Challenging Students: Reflections on the Development and Delivery of an Undergraduate Module That Introduces the Full Systems Development Life Cycle

Steve McRobb
Faculty of Computing Sciences and Engineering
De Montfort University
Leicester LE1 9BH United Kingdom
smcrobb@dmu.ac.uk

ABSTRACT

This paper reflects on the experience of developing and teaching an innovative, experimental undergraduate module in Information Systems Development. The module aims to give first year students a rounded experience of systems development from feasibility to evaluation. Students produce a series of analysis and design products that lead, finally, to the implementation of a distributed 3-tier web-based prototype system. Many staff regard this as overambitious, since most of the students are completely new to systems analysis and design and are concurrently learning the rudiments of programming and web development in other modules. The paper discusses the institutional and educational pressures that led to the conception and development of such a demanding module. It describes the process of negotiation and compromise through which the module came into being. And it explains the support mechanisms that have been developed to make it possible for students to succeed. Results are presented that indicate the module succeeds in several ways. It lays a useful practical foundation for later studies and work. It gives scope and encouragement for abler students to excel. And its support mechanisms help weaker students to exceed their own expectations by acquiring skills and understanding that they think at first are beyond their reach. The paper closes by summarizing the key lessons learned by the author, which include insights into the use of scaffolding and formative assessment to motivate students, and a greater willingness to experiment.

Keywords: Project, Methodology, Systems Analysis and Design, Systems Development

1. INTRODUCTION

This paper describes the commissioning, development and delivery of a module called INFO1401 'Information Systems Development' – a first year undergraduate module delivered as a core part of several degree courses at De Montfort University, UK. (De Montfort, or DMU, is a large institution geographically at the centre of England, and also near the middle of the UK University league tables). The history of this module illustrates many of the pressures and constraints that affect the teaching of systems development today. It is also innovative and experimental in a number of ways.

The paper has some of the character of a research report, and includes some elements of empirical research. But what follows was not planned as research and followed no proper methodology. Some statistical analysis is presented, but this is really intended only to support some general points about pass rates and grade profiles.

The pedagogic approach of the module can be rooted in the constructivist literature. For example, the teaching and assessment rationale has a clear basis in Vygostsky, especially the "scaffolding" of learning (Vygostsky, 1978). The module's assessment strategy also has strong links with Kolb's learning cycle (Kolb, 1984) as it encourages students to move from practical experience and experimentation to reflection and conceptualization. However, the discussion that follows will make no further explicit reference to theory. Most of the work was simply undertaken as a normal part of the author's daily duties as a lecturer, course leader and module developer, and the paper has been written as a reflective report on practice, not a theoretical discussion. Regardless of its classification, it is hoped that readers will gain useful insights from the story.

INFO1401 seeks to convey a rounded experience of information systems analysis, design, implementation and evaluation, using an extended case study and practical lab work as the basis for a series of group coursework deliverables that reflect typical lifecycle phase products. Alongside this, a traditional series of lectures and tutorials explains the context and the techniques of systems

development, and gives students the chance to practice and get feedback on their work.

Much of the delivery is conventional, especially the early coverage of information systems, project lifecycles, project management, analysis and design techniques and so on. But two aspects are quite experimental, and these are the main focus of the paper. First, there is the distributed web architecture of the software prototype that students design and build. Second, there is the assessment process. Key points to emphasize are that the module challenges students by setting them tasks that some consider far too ambitious for first year students, and that it makes creative use of assessment to support students through the most challenging of these tasks.

INFO1401 is a key module, whose role is to integrate a student's experience of his or her course. It tries to fit techniques acquired on other modules into a coherent approach to systems development. Teaching systems analysis and design is anyway a difficult task. There are complex interdependencies between the various elements, and where more than one person is involved, a coherent and reasonably consistent approach must be found that reflects the skills and interests of the participants. This paper documents how a small team negotiated their way past obstacles and through the many compromises encountered in an attempt to find a new way of doing this. Two key themes stand out from the experience, each a dialectic of opposing tensions.

One of these is risk *versus* caution. Students rise to a challenge, but need appropriate support and safety-nets if they are to overcome the difficulties. Teachers also need support mechanisms, although we usually have the privilege of designing our own. Our success, too, depends on balancing risk against caution. The second is technical depth *versus* holistic thinking. While these are sometimes rallying points for two distinct worldviews, and can even seem to be mutually exclusive, both are necessary in systems analysis and design.

For INFO1401 to succeed, it was necessary to resolve, even to transcend, both of these dialectics.

The paper is structured as follows. First, it briefly describes some of the institutional background. This is relevant as the module explicitly seeks to address at a small scale some problems that led to a university-wide review of the undergraduate curriculum. At the macro level there were both pedagogic and political factors that affected the shaping of INFO1401. Next, the paper discusses the early definition of the module (part of its 'template' – an outline statement of aims and content – is included as Appendix 1). Here, some local (i.e. departmental) political factors came into play, while pedagogical issues were examined in closer detail. The paper then covers the implementation of the module, followed by the experience of running it through two successive deliveries. Here, the pedagogic focus zooms in to the specific mechanisms by which the module's aims are fulfilled in practice, and the political focus shifts to a more pragmatic, personal level. Student and staff emails and discussion postings are used to illustrate some issues (the available material is too thin for this to be really representative, but the comments are illuminating nevertheless). Finally, the experience is reviewed, some conclusions drawn and some recommendations made for future practice and for further investigation.

2. SMALL BOAT IN A STORM: LIFE AT A HIGHER EDUCATION INSTITUTION

The story began very early in 2002 when central management announced a radical, institution-wide process of change. Like many UK universities, DMU moved in the early 1990s to a two-semester delivery pattern combined with a flexible modular degree scheme. During the next decade dissatisfactions mounted, and it was now proposed to switch all undergraduate teaching back to year-long modules 'owned' by their courses, rather than completely independent of them, as had become the norm.

Modularity was widely blamed for causing students' conception of their studies to become fragmented, and the new framework was intended to change this for the better. The proposed change was expected to mean more than just a structural change to the delivery pattern. Every module must be reviewed, even if this was only to see how it could be best delivered in a longer mode. But the temptation to review and update the entire curriculum would be hard to resist. As a result, the change process soon became known – rather misleadingly – as 'Curriculum 2004.'

The extension of modules to a 24-week delivery pattern was expected to enable students to develop their skills and understanding in a more measured way, and to give more time for reflection and recapitulation. Also, since courses would have more or less exclusive ownership of their constituent modules, it would be easier to maintain subject integrity and coherence.

A report to Academic Board explained the benefits thus:

> "...the curriculum re-write would provide an ideal opportunity for course teams to thoroughly overhaul the curriculum. The use of year long modules provide [sic] scope for imaginative and creative approaches to course design in terms of flexibility within modules, greater use of formative assessment, study in depth and an overall increase in course cohesion which it was felt should aid student retention and enhance student satisfaction." (DMU, 2002)

All of this has very direct relevance to the particular aims that would eventually be expressed as INFO1401, the module under discussion. Staff in the School of Computing generally welcomed the opportunity for a comprehensive review of their subject material. New technologies, new programming languages, new analysis and design methodologies and notations had emerged, and some modules had not kept pace. Changing employment patterns were also a factor. From placement visits, from anecdotal evidence that circulated around the School, and from a variety of other sources, it was understood that the role of the systems developer had changed. The construction of isolated

single systems was (and is still) being supplanted by an emphasis on systems integration, the assembly of software components and the web-enablement of existing legacy systems.

Curriculum 2004 was seen as providing both the opportunity and the political momentum needed to correct the drift towards irrelevance. Over the following year, a group of subject and course leaders produced new course designs and outline specifications for new modules. As course leader for the Business Information Systems (BIS) degree course, the author was a member of this team. Module outlines were then handed to module leaders for detailed design, production of materials, and in due course for teaching and assessing the students. By this point, the author was also module leader for INFO1401, and thus had a uniquely privileged view of the whole process. Figure 1 shows the key stages.

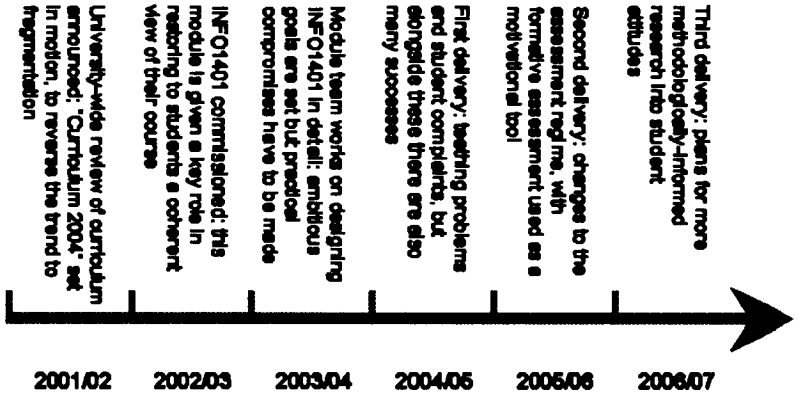

Figure 1: Timeline for the commissioning, development and delivery of INFO1401.

The links between modules – both horizontal and vertical – had become seriously eroded during the University's decade of modularity. This was compounded by the loss from most courses of 'integrative activities' (group projects typically run outside the normal timetable and aimed at integrating the knowledge and skills of several modules). But there was some disagreement, partly cultural in origin, about exactly what problems our new curriculum should be addressing.

The School of Computing has historically been split into two loose staff groupings, which cleave roughly along the boundary between the former departments of Computer Science and Information Systems. Neither department exists any longer, but this polarization of views can still be discerned.

We will call the groups *Technicos* and *Informaticos* and for simplicity, and at the risk of some over-simplifcation, we will characterise them as follows (most readers will recognise some of the cultural and political composition of their own faculties).

Technicos generally emphasise technical skills and precision, and might define the key aims of a computing degree course as: to equip students to formulate a clear and unambiguous statement of a problem, to specify a technological solution to that problem, and to use the most appropriate technology available to implement that solution in an effective way. DMU's *Technicos* saw our key academic problem as a decline in students' technical ability, especially their programming and networking skills. The solution was a return to intellectual, especially mathematical, foundations; for example, more emphasis on programming, algorithms and data structures, and a partial restoration of structured approaches to analysis and design.

Informaticos agree up to a point on the need for intellectual and technical rigor. But they also take a wider view of the nature of systems, and thus of systems development. Success is seen as critically dependant on a deep understanding of the needs of users, clients and other stakeholders of a system, both those that are given explicitly and those that are implicit in the context. Where this understanding is missing, inappropriate systems will be developed regardless of the developer's technical skills. Students need to integrate their knowledge and diverse skills within an overall framework. This ties in with some skills and knowledge that employers value in all graduates, irrespective of discipline: project management, understanding of project lifecycles, project deliverables, group working, communication and research skills, and systems analysis and design approaches.

Viewing the curriculum design problem from this perspective, a meeting of the Information Systems subject team noted that:

> "Certain problems with the current curriculum were identified; these tend to manifest themselves in the final year (and via placement employer feedback), for example final year students across all Computing courses who:
>
> - "are unable to make a coherent link between the different parts of the systems lifecycle;

- "are unable to construct documents such as a Feasibility Study, Requirements Specification, Design Document;
- "know nothing about research skills;
- "and who are therefore insufficiently equipped to approach their final year project, never mind professional practice!" (Prior, 2002)

The strategy proposed to address this was a new first year module called "Systems Development":

> "Covering the whole lifecycle from project initiation, through analysis, design, to implementation. Students to acquire an overview of the entire lifecycle, to understand key deliverables such as Feasibility Study, Requirements Specification, Design Specification and learn an approach to project management. A generic approach to be taken.
>
> "[The new module was to be] Core for all undergraduate [computing] courses" (ibid.)

By September 2002, the BIS course team had begun work on the proposal. The following were agreed as key learning outcomes for the first year of the redesigned course:

- "Strong grasp of systems analysis and design method and techniques.
- "Basic skills of practical systems development
- "Fundamentals of understanding businesses and organizations" (McRobb, 2002)
-

INFO1401 would play a role in meeting all three aims, but with primary responsibility for the first. Other modules would focus on programming and business topics such as accounting and marketing, but INFO1401 would integrate the student's understanding of the course as a whole.

At the same time, it was expected to fulfill a similar role on several other courses including Computer Science, Software Engineering and Computing Joint degrees. A report to the Subject Authority Board described the module as follows:

> "The module INFO1401, Information Systems Development, will present an integrated view of the various stages and techniques of the systems development lifecycle, and will also develop an understanding of the application of modern information systems within organizations." (McRobb, 2003)

From an *Informatico* perspective, these were central aims for the entire School. But the use of this module within other courses was to profoundly complicate matters. Curriculum 2004 was understood to offer a simple relationship – that a course would 'own' its modules – but INFO1401 seemed unlikely to benefit from this simplicity. Next, we will examine how the grand scheme was translated into practical content, delivery and assessment.

3. TAKING ON A LIFE OF ITS OWN: HOW THE DESIGN EVOLVED

Detailed design and development of INFO1401 occurred mainly in the summer of 2004. This was quite late, given that the first run was due to begin in September, but it was also unavoidable as the module team was only established at the start of the summer. This team consisted of two *Technicos* led by the author (an *Informatico*, in case the reader has not already guessed). The module learning outcomes were now defined as:

> "1. Explain key concepts in the Information Systems domain, and discuss the impact of IS on individuals, organizations and society.
> "2. Explain the role, significance and typical activities of project selection, project management, systems analysis and design.
> "3. Apply appropriate techniques to produce a requirements specification and design for a constrained case study, based on supplied information about user requirements.
> "4. Apply practical systems development skills to implement a prototype system in an environment such as MS Access.
> "5. Evaluate the extent to which the implemented system satisfies user requirements." (DMU, 2003)

In addition, the module template stated that:

> "...a key emphasis throughout will be on the provision of practical examples and the opportunity for students to undertake practical project initiation, systems analysis, design and implementation tasks" (ibid.)

The module content of the various degree courses had now mostly stabilized, which meant that most constraints and requirements seemed reasonably clear. Almost all students on INFO1401 were expected to cover the foundations of HCI in another module. All were expected to take a programming module. Later, all would study object-oriented analysis and design, entity-relationship modeling and SQL. INFO1401 was to adopt a structured lifecycle, with a toolkit of dataflow diagrams, entity relationship models and structured English. Thus, by graduation, all students would have some familiarity with structured, object-oriented and data-centered approaches to analysis and design.

MS Access was initially assumed to be the vehicle to introduce database concepts and SQL, and it had a number of clear advantages. Students could develop practical skills useful to placement employers, an important point as most were expected to spend a year on work placement. Even weak students should be able to achieve some satisfactory results, an important point since many students undertaking this module would be new to computing and concurrently learning programming for the first time.

However, this logic shifted as horizontal links with other new modules began to exert strain. In the early stages of curriculum redesign, courses and modules co-evolved in an iterative manner. But as module content grew more detailed, there was no time for further iterations. Conflicts that emerged at this stage could not always be resolved through mutual interaction between modules, nor could gaps be so easily filled. Sometimes one module just had to adapt to the changes in another.

It emerged that the HCI module to be taken by most students had shifted focus to web page design and implementation.

INFO1401 must either work with this or fit some HCI into an already crowded syllabus. While the former option was more attractive for the schedule, it also slightly weakened the argument in favor of Access.

Another issue was the parallel evolution of the Software Engineering course. No other module on this course now covered E-R modeling and SQL. The course leader sought assurance that INFO1401 would teach his students to design and create a database and to query it using SQL. The ease of use of Access, previously seen as an advantage, began to seem a positive *disadvantage*. Built-in wizards, query-by-example and so on were attractive when the aim was for students to implement their designs without too much fuss. As SQL assumed more importance, it became less desirable to use a package that would allow students to implement and query their database without writing or even understanding a single SQL statement.

Programming was another complication. Access uses VBA, which is essentially a subset of Visual Basic. Some students (those on BIS and Computing Joint courses) would learn VB.NET, but others (those on Computer Science and Software Engineering) would only know C. This caused a dilemma. The need for coding could be minimized, but the prototype might then become too trivial. Some class coverage of VBA could be added, but the schedule was already tight. Students could locate their prototype functionality in separate programs that used the Access database only for data storage, but this would increase the overall complexity and would also required extra class content. All solutions led to further complications.

The final argument against Access was its specificity to Windows. An industrial strength RDBMS such as Oracle or MySQL would have more relevance to placement and graduate employment. As it happens, both are installed on Unix servers, and some exposure to Unix or Linux would also help to provide practical experience relevant to future employment.

At this point, one of the module team (a *Technico*, needless to say) proposed that we base the students' prototypes on distributed, three-tiered web architecture. We could then employ a web interface and either MySQL or Oracle RDBMS as the database layer. The lack of a common programming language remained a difficulty, but our *Technico* was also a CGI expert and argued this could readily be adapted to use either C or VB as its host language. First year students would certainly face significant difficulties in undertaking such a complex implementation on their own. But this could be overcome by the release of template code for the harder parts of the task. For example, CGI code could be issued to achieve integration with the database layer. The students would be expected to understand this code and to adapt it for their own use, but not to write it from scratch.

There was no consensus within the team. The other *Technico* disagreed, partly due to his lack of experience with CGI, but also because he believed a distributed web architecture would be too difficult for the students to master. The winning argument for the author was the strong relevance to future employment. We could offer our students an exciting and truly groundbreaking experience. Not only would they cover the entire lifecycle, but also their products would include original software development and the configuration and integration of software components. In the course of this, they would gain experience of a range of technologies not usually introduced until the second year, or more usually the final year, of undergraduate study.

The risks were equally obvious. Few students would already have the programming skills, and it was both inappropriate and infeasible to consider teaching these within the module. There must be a great deal of support to ensure a reasonable chance of success. Code templates would be needed for any aspect of the middle layer likely to be too difficult for novice programmers. Nor could the students be expected to design, never mind implement, a robust, functional database without significant assistance. There would be gaps in their HTML skills, too. The prototype architecture would critically depend on forms for data input, and again the choice was either to teach the skills or to provide templates of some sort.

The dissenting tutor was out-voted, and the decision taken to proceed with the web architecture. The ethos of the module team changed perceptibly with this decision. Early meetings had concentrated on subject content and the teaching plan. Now the focus on group coursework intensified. Preparation of lecture and tutorial materials continued, but became a background activity. Meetings emphasized technical details of the implementation, and the support mechanisms that the students would need.

Oracle was selected as the database, partly for its familiarity and partly because for the simple web interface to its interactive SQL module, allowing us to hide the rest of Oracle's complexity. The C programmers would use compiled C/CGI programs running on a CGI server for their middle layer, while the VB group would use an ASP server to execute VBscript embedded in HTML pages. The user interface layer would be written in HTML, with forms as a means of data input and tables to control page layout and to format data output.

The team approached the planning in an atmosphere of excitement and enthusiasm that probably contributed to our biggest problems, although this would not be apparent until some months later. It is always risky to set a challenge for students that you do not *know* you could meet yourself. When the distributed architecture was first proposed, our skeptical tutor urged the construction of a working prototype to prove the concept. This was noted as an urgent priority at a team meeting three weeks before start of term, but it was not completed until after the students had begun work on their own prototypes. One reason was lack of time and pressure of other work. But more importantly, neither the CGI nor the ASP server, on which the whole architecture would depend, was yet operational. Our sceptic now urged that we fall back to a safer development path, and rewrite the coursework for MS Access. But pressure of time (again), the amount of extra work involved in making the change and the

confident assurances of our CGI expert all argued against it. As module leader, the author decided to proceed as planned.

4. FIRST STEPS: THE 2004/2005 DELIVERY

The group coursework required students to submit a feasibility report, a requirements specification, a design specification and finally to implement a software prototype. Students would require different types of support at each stage, and for varying reasons.

Clearly, students must be able to get back on board if they missed a phase for any reason. We had also to ensure that students' misunderstanding of earlier tasks did not handicap them in their approach to later ones. Indicative solutions were to be released at the end of each interim phase via a password-protected website, limiting access to groups who had already submitted their own version. Each solution would be a reliable foundation for work on the next phase.
To manage the level of difficulty for the students in the coding phase, undoubtedly the hardest part of the project, a range of aids were eventually provided.

- Sample web pages illustrated how forms can link to a CGI program.
- Code samples in C/CGI and VBscript illustrated, separately for each language group, how to read data from a form, how to insert variables into SQL queries, how to send the query to the RDBMS, and how to format the output into a new web page.
- A thoroughly tested data model, and an SQL script to create the corresponding tables and to populate them with sample data.
- Example SQL statements that executed similar functions on a different data model of comparable complexity.

Further code samples were also released on the module Blackboard site, and students were encouraged to adapt all supplied code to their own designs. Much of this was also taught in lab classes, so that all students were expected to be able to contribute to their group's work.

However, our incomplete summer preparations soon began to have a direct impact. First, there were serious delays in getting the CGI server operational. Technical and teaching staff put in many extra hours to get the server running, but classes that explained what to do during the implementation phase had to be delayed up to two weeks. This was a frustrating time for students. One group registered their concerns in an email to their course leader:

> "We are currently doing part of the unit which requires us to code and understand cgi and sql yet most people feel that we aren't actually being taut this and just being told to do it...
> "Not sure what you can do about these problems but I'm guessing you have already said something as today's lecture started to explain how cgi and sql work" ("INFO 1401" – email to course leader, 2005).

In his response to the students' difficulties, our skeptic commented:

> "I'm quite sure that a lot of my students are having the same sorts of problems with CGI etc... However I think SQL is fine at least for the students who've done the work. But coping with the mix of environments is still a problem, and the students probably think that they're not getting enough in lectures to really understand stuff they need to use. I think this problem is inherent in... the aim of getting students to create running systems with a three layer architecture" ("Course Problems" – email to the author, 2005).

Others posted their views on the Blackboard feedback forum. One wrote:

> "I followed instructions and turned up to the lab to find not only was the cgi link down but the server not functioning properly Still!!" ("This module is Ridiculous!!" – email to Blackboard feedback forum, 2005).

Installing and using the C/CGI utilities and the Oracle CGI pre-compiler also proved more difficult than anticipated. As a result, the supplied code material went through a series of revisions. This was a further source of frustration:

> "We were given things to aid us in completing our database prototype but they have been wrought [sic] with problems and have had to be revised... there's a 'Makefile' which compiles the pro*c and this has been revised three time, up to last week. As this file is essential to doing anything to our database it has been rather difficult" ("Concerning INFO1401" – email to Blackboard feedback forum, 2005).

Coding and implementation was much more difficult for the C/CGI group than for their ASP/VBscript colleagues, and only strong programmers in the former group produced really successful prototypes. But by the time these problems were apparent, we were too far down this road to change. A wholesale switch to ASP was considered at one point, but this would have meant writing off the time already spent explaining the C/CGI approach, and it also risked confusing many students even further.

Instead, our CGI expert gave a lot of extra, unscheduled support in the labs. An announcement was posted to let the students know that the marking would take into account their difficulties; credit would be given for what was attempted, even if not fully successful. In any case, for some weeks before the final deadline laboratory classes were devoted to supporting the groups as they worked on their prototypes. Email support was almost around the clock in the closing stages, and a lot of tutor effort was expended on debugging student code.

In the end, the pass rate and average marks for both C/CGI and ASP students were broadly comparable to other modules. But this was largely because the prototype deliverable represented only 10% of the final mark. An analysis of the final results for this run shows that the average exam mark was just 2% lower than the average overall coursework mark. This difference in marks is not untypical for a module in this general subject area. But a comparison of the coursework component marks is more

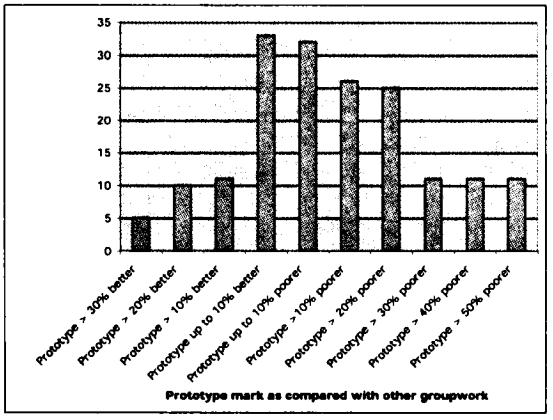

Figure 2: Comparison of prototype mark with combined mark for other group work (2004/2005 cohort).

revealing. The average student scored 8% less for their prototype than for their feasibility study, analysis specification and design specification combined. Two thirds of all students performed worse on the prototype, with only one third performing better on this task. The variations between marks for the prototype and marks for the rest of the coursework are illustrated in Figure 2.

This was certainly not enough to restore goodwill among these students. It also brought a political defeat that in turn undermined a key purpose for which the module was originally proposed. During this difficult period, the course leader for the C programmers, a leading *Technico*, responded to their complaints by getting approval to drop the module from their courses for the following year, before its first run was even complete. Originally designed to ensure that technical and contextual subjects were integrated right across the curriculum, the module had lost this role on two of the School's main technical courses.

5. MAKING STRIDES: THE 2005/2006 DELIVERY

For the second run of the module, several changes were made. The teaching schedule was adjusted to reduce the time spent on theory in the early weeks, and to permit an earlier start on the practical tasks that had caused the greatest difficulty in the first run. Since no C programmers would now be taking the module, only ASP need be used for the prototype middle layer and CGI was dropped. Perhaps more significantly, the assessment was changed to a blend of formative and summative strategies that has proved popular with students. This was prompted, and most of the details worked out, at a summer staff development workshop (Mortiboys, 2005).

All group deliverables except the final software prototype – feasibility report, analysis specification and design specification – are now submitted first in draft form. At the end of each phase, groups attend a review meeting with the tutor. An indicative mark is given together with feedback on how the work could be improved. If a group wishes to improve their mark, they can so up to the final deadline when a portfolio is submitted with final versions of all interim products and also the software prototype.

This procedure does not exactly mimic the real world, since a real team would not be permitted to continue to the next phase if the products of the current phase were unsatisfactory. But the opportunity to revisit and revise earlier work is appropriate for an educational context and helps to teach students how lifecycle products are related.

Submission of draft work is mandatory. If any group does not submit a draft, they forfeit the corresponding mark in the final portfolio. Students are rewarded (by feedback, and also by maintaining the chance of a better mark later) for submitting *something* for review, no matter how poor its quality. If no submission is made, they are punished by the irretrievable loss of marks. Students who lose early marks through non-submission of draft work are also reminded of the greater need to meet later deadlines, since they must now compensate for the lost mark.

At the time of writing, the marking for this cohort is not yet complete, so it is too early to make a comprehensive assessment. But it is already clear that many groups have made strategic decisions to improve early work for which they received poor feedback. This has encouraged them to maintain their efforts through what is often a low point in the academic year, as everyone's energy began to flag during the run-up to the Easter vacation. The author has also observed several students – and some whole groups – spurred into action by the realization that they will soon lose marks if they do not make a submission. The review meeting then provides an opportunity to discuss difficulties and misunderstandings that might not otherwise have been

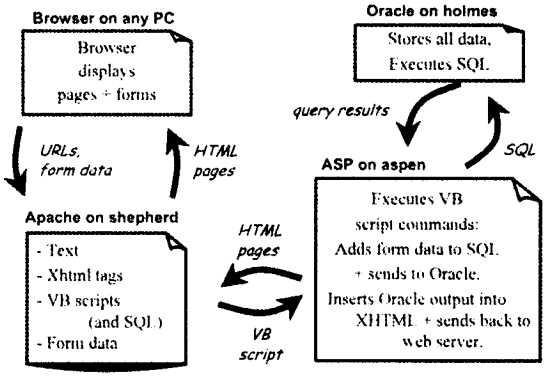

Figure 3: The prototype architecture.

picked up by the tutor until it was too late to recover.

A further piece of individual coursework has been added to the mix. After all the group work is complete, students are asked to write a reflective report. The assignment specification asks students:

> "...to reflect critically on one aspect of the group coursework. This should be something that is (or was at the time) important to you. It could be positive or something negative. Maybe you achieved something that makes you feel proud. Or maybe a problem

occurred and you think it could have been handled better. The important thing is that you propose a topic that **you** want to write about." (DMU, 2005)

Many of the resulting reports give insight into student attitudes and some selections are quoted in the following sections of the paper.

6. THE WEAK GO TO THE WALL, BUT THE BRAVE ARE CHALLENGED

In an echo of the way that weak or infirm worshippers in mediaeval British churches (which lacked seating) were encouraged to "go to the wall" for support, INFO1401 has "walls" built in to support students who need them. These are necessary because the coursework is both demanding and protracted. It is demanding because it relies on the application of many skills, most of which have been covered only briefly in class. It is protracted because it involves a series of deliverables, phased through the greater part of the academic year, with each to some extent a direct development from, or derivative of, its predecessor. It was anticipated that some students would need a great deal of support if any significant learning were to be achieved. The assessment strategy contributed to this, and differing approaches have been tried in successive years, as described earlier.

It would be misleading to think that students now found either the module or its coursework easy as a result of all the support. Results and informal feedback (mainly oral feedback in class, and unfortunately thus not formally documented in any way) confirm that it is still regarded as a difficult task. The support mechanisms helped most students to achieve a reasonable result under difficult conditions. But a small minority of groups still completely failed to build a functional prototype, despite all the help on offer. Many said that they found the module as a whole difficult in comparison with their other first year modules. Following the second run, one student commented on Blackboard that the module:

> "covered many aspects of information systems that i did not expect in the first year... [but] the overall module content was good [and] had good depth" ("Module Content", email to Blackboard feedback forum, 2006).

The difficulty was more pronounced during the first run, when the module was still in its own learning phase. But even during the second run the final (software development) phase of the group work remained a serious challenge. Several students asked the author to confirm that it really *was* a level one module. This echoes the open disbelief of some staff when they hear what the module asks its students to do.

In general, the feasibility, analysis and design stages were completed with no more than the usual difficulty. But many frustrating pitfalls lay in wait during the coding phase. One of the trials for novice programmers is the realization that apparently insignificant errors can cause a function to fail completely. There are many places where such errors can occur. Figure 3 shows the main components in the architecture.

Most students had little or no experience in debugging, and it took time to learn this, particularly in such a complex environment. It was common for the author to be asked why they had to use HTML, ASP and Oracle instead of Access. His litany in reply – that this experience would ultimately be extremely useful for future employment, seemed to sink in. One student – from a group whose final prototype showed no working functionality – wrote that:

> "The skills I have developed during this coursework can be very useful for any person involved in an information technology related job in the future. Many people can easily create a database in Microsoft Excel or Access however having the ability to create a system using the Oracle database would be very beneficial" (Assessed Critical Report, 2006).

7. DIAMONDS IN THE MUD

Final prototypes fell along a wide spectrum of achievement. At one end, a handful consisted only of one or two web pages, with weak understanding of the basic requirements, very poor design, no functionality and little evidence in the source code of any real grasp of the technical aspects of the project. At the other extreme, some elegant, sophisticated prototypes showed excellent design skills combined with a clear understanding of the requirements, and also signs that the feedback given on earlier deliverables had been taken into account. The majority of groups achieved a great deal, despite the many difficulties they had to overcome.

The aim was to foster confidence across the full range of achievement, and there is some evidence that this occurred. During the first run, some very limited research was carried out using a questionnaire instrument administered to a small sample of students, chosen to represent a wide range of ability. Four of five respondents stated that their confidence increased as a result of the module; two of five indicated that it had increased a lot. One wrote that the module as a whole was:

> "a positive experience which has made me aware of the issues involved in the process of Systems Development" (Survey response, 2005).

Critical reports completed by all students during the second run give further confirmation. One student, who was suspected by the author of being a rather passive member of a successful group, reported that his experience:

> "helped me to develop my communication skills and put my ideas across. It also helped me to adapt to criticism and improve on failed attempts of the work... I have become more determined to improve areas of my work that need to be improved and gain satisfaction out of this" (Assessed Critical Report, 2006).

Another wrote that he was:

> "especially pleased with our final prototype that we created, this was very much a team effort, it was a challenging task yet through many hours of persistence and supporting each other we managed to create a

> working basic model that left us with a sense of achievement" (Assessed Critical Report, 2006).

Some prototypes exceeded the stated requirements in completely unexpected and imaginative ways. For example, one group used real ISBNs for books in their database (the first case study was a University library) and linked these to images on Amazon.com. Their application was thus able to display the correct cover image for each book. Another group found an ingenious way to generate a primary key for a new book copy. Their script retrieved current copies from the database, counted the number of rows in the result table, incremented this and then inserted it into the new primary key. In Access this is a built-in function and would be quite trivial, but in this environment, and for students with such limited programming experience, it showed real initiative and problem-solving ability.

Several groups implemented a log-on process, although this was not a stated requirement. For it to work, the database definition (supplied as an SQL script) had to be modified to add the new table needed for user data. The log-on process itself is quite complex in comparison to other functions, since there are several exception conditions along with the successful log-on scenario. One group even managed to implement session keys, which they used to track the users who were logged on at a given time.

These are exceptions, and not the norm for the module. But they are closely trailed by many lesser achievements that are still significant in their own right. Other groups were narrowly defeated, often just by lack of time, in their attempts to implement more complex functions such as multi-table queries (table joins were barely mentioned in the brief lecture coverage of SQL).

One group declared, at the start of the implementation phase, that they had no confidence that they would ever manage to make all the layers work together. Yet, just a few weeks later, they couldn't stop grinning as they demonstrated the results, which far surpassed their own expectations. This group's prototype was functional in all the required ways; they had also discovered an elegant way of using a single page as both input and output screen (by including the code for the input page within the script file, which then recursively called itself again with each new input). Another student reported that:

> "the most encouraging part was getting the feedback from the project, as it then made us all realise that the hard work that we had put in, was worth every single minute of it" (Assessed Critical Report, 2006).

8. DISCUSSION

INFO1401 arose from a combination of circumstances. At the very beginning, there was a perceived need to address a problem in current operations and a climate in the organization that was amenable to change. There was also disagreement among colleagues about the nature of the problem and the best solution. The eventual composition of the module team had a profound influence on the implementation of the module. Without a radical *Technico* input, the author would almost certainly have followed a much safer path. Without the strong *Informatico* influence exerted by the author, above all in the explicit links and frequent signposting, both forwards and backwards, between different phases, tasks and products, the students might only have learned a little each of a lot of different technologies. For some, at least, the module worked as it was meant to do. One student summarized his experience as follows:

> "I learned the techniques and understand [sic] of how systems life cycle develops from one stage to another and all the investigation and analysis that is required to lead up to a successful system" (Survey response, 2005).

This is not a bad paraphrase of the original module aim quoted earlier in the paper. It can be argued that the success of the module is due to a synthesis of holistic thinking with real technical challenge. On a similar note, one of the small bands of Joint degree students (technically speaking, the least prepared of all) wrote:

> "I think it is important to have a module like this on the first year as it shows the full picture of Systems Development from the beginning of the degree" ("INFO1401 questionnaire" – personal email, 2005).

There was another dialectic at work, between risk and caution. This is visible in the polarization of staff views over the distributed architecture proposal. But the same theme recurs throughout the module; it is there in the balance that was sought between challenging students and supporting them. Environmental factors, and especially developments in the employment market, added their weight to the argument for risk. Educational factors, and a general background of concern about falling achievements, argued for prudence. Here, too, the success of the module can be seen as depending on a synthesis of contrasting elements.

In simple quantitative terms, the module is certainly successful. In its first run, 185 students from three single-subject degree courses and a variety of joint degrees were enrolled. In background and in inclination, the BIS students were among the least technical of those who took the module that year, and many saw it as very demanding and very technical in nature. Yet the first attempt pass rate for this group was 74% – equal highest among their computing modules.

The results for all cohorts together indicate a healthy module, with an overall pass rate of 81% and a top mark of 89.4%. The marks follow a slightly skewed normal distribution, centered roughly in the lower C grade, as shown in Figure 4.

The most striking risk was the decision to adopt untried technology and some students were definitely alienated by what they saw as the unreasonable demands that were made on them as a result. But overall student performance was not adversely affected, and perhaps the module would not have achieved its other successes had staff not been willing to take this risk.

The module's greatest failure – being dropped from two leading technical courses – arose directly from this same

decision. Yet still, albeit for a narrower range of students, INFO1401 does help students to understand how information systems can be produced that are related to the needs of its users and their context of use. The module achieves relevance to real world systems development by focusing on systems integration rather than development *de novo*. And student participation is clearly enhanced by the encouragement to rework interim products (only one team out of 20 did not take up this opportunity). One student whose group submitted almost nothing for the earlier phases, then worked hard to catch up before the final deadline, reported that:

> "once we actually worked together as a group and managed to concentrate on the coursework we realised that the work was not as hard as first thought and could have been completed for the sectional deadlines" (Assessed Critical Report, 2006).

9. CONCLUSIONS

This paper has presented a necessarily incomplete history of the development and first two deliveries of an innovative module that seeks to introduce first year undergraduates to the full life cycle of contemporary information systems development. Its assessment relies partly on students producing a series of deliverables that are chosen to reflect the process of developing a real system. Feedback is given on interim products, and students can then improve their work before final submission. While none of these assessment ingredients are original in themselves (although the resubmission of formative work is relatively unusual), it is likely that this particular mix is unique.

The nature of the tasks the students are asked to undertake is more clearly innovative. Some modules at DMU and other Universities superficially resemble INFO1401 in their content. But these generally focus on technologies rather than on the life cycle itself and are studied in the second or final year rather than the first year. INFO1401 is, as far as the author is aware, unique in using a complex software environment as a vehicle to teach the overall framework of systems development to students who are just beginning their degrees.

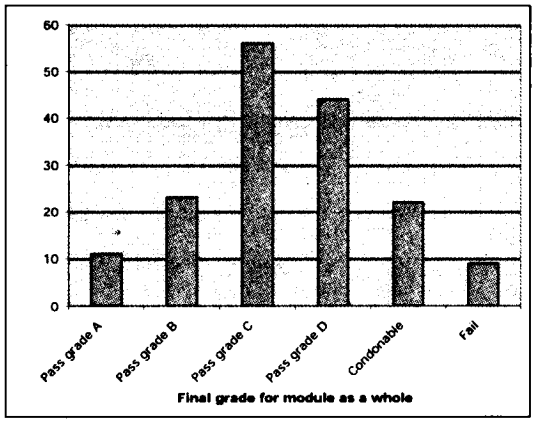

Figure 4: Overall module results (2004/2005 cohort).

The point is to convey a holistic understanding, not to achieve any particular level of technical skill, although certainly many useful practical skills are acquired along the way. It is likely, due to the changing role of IS professionals and also the continuing evolution of software technologies at work, that such approaches will become commonplace. In reflecting on the experience so far, some general conclusions can be drawn.

- Students are strongly motivated by the opportunity to rework interim products following tutor feedback, although this needs to be offered within clearly defined parameters.
- Students who are given appropriate support can meet challenges that they, and many of their tutors, believe at first to be beyond their reach.
- Student perceptions of the difficulty of a module are not necessarily reflected in their levels of achievement.
- It is possible to integrate technical challenge with a holistic understanding of the framework within which the technologies are embedded.

The learning curve has at times been steep, both for students and for their tutors. One of the greatest costs has been the sheer amount of staff input required, and the level of active intervention by staff that was needed, particularly during the prototype development phase, was not properly anticipated for the first delivery. As a result, the teaching schedule for the second delivery was adjusted to increase the amount of time available for this phase, both in terms of calendar time and staff contact, particularly in laboratory classes.

As the module is reviewed for its third delivery, one aim is to reduce the demand on his the author's own time, provided that this can be done without sacrificing the quality of learning. This may be achieved through further experimentation with the nature and timing of the assessment tasks.

A number of aspects of the module merit further research. In particular, it is planned that during the next delivery, all students will be surveyed at certain key points to investigate whether, and how, their understanding and confidence change during the year. It is the author's hypothesis, not as yet formally tested, that the module's balance of risk and security is effective at enhancing its students' independence and self-confidence. (Such a survey was in fact planned for the second delivery, but due to other pressures it was not possible to implement it in time).

Perhaps the most important lesson is that a tutor should not be scared to take her students to places she has not yet thoroughly explored herself. A tutor's role is to help students learn from their experiences, and this does not require complete control over every aspect of the situation at the outset. Related to this is the well-established point that useful learning occurs when students are actively engaged, preferably in a self-directed task, rather than being 'taught' something by a tutor. This module offers one way of achieving this desirable state of affairs. However, while the students' work for INFO1401 is largely self-directed, there is a heavy demand on staff time at certain stages, and the tutor

must be willing to respond. It is also essential for a module to have a clearly thought-out plan that links back to the strategic aims of the course – in exactly the way that a requirements specification defines what a system should do without tying the developers down in specific details of how it will be implemented. Where such a framework exists, it is easier to be flexible regarding the detailed implementation and the day-to-day running of the module.

10. ACKNOWLEDGEMENTS

The author wishes to express his gratitude to the many colleagues who advised on and helped with the development, teaching and review of INFO1401. In particular, Jordan Dimitrov, Mike Leigh, Mary Prior and Martin Stacey all deserve special mention. Grateful acknowledgement is also made to Ralph Birkenhead and Bernd Carsten Stahl, without whose encouragement this paper would not have been written. Finally, the encouraging comments of the paper's anonymous reviewers are greatly appreciated, and their suggestions have been accommodated wherever possible.

11. REFERENCES

DMU (2002), "Joint Meeting to discuss the implementation of the Academic Calendar Proposal: Summary Of Recommendations To Academic Board", DMU internal document, 29 January 2002.

DMU (2003), "INFO1401 Module Template", DMU internal document, 2 April 2003.

DMU (2005), "INFO1401 Critical Report Coursework Specification", DMU internal document, 27 January 2006.

Gifford, B.R. and Enyedy, N.D. (1999) "Activity Centred Design: Towards a Theoretical Framework for CSCL", Proc CSCL 1999, Stanford University, Palo Alto, CA, Hoadley, C. and Roschelle, J. (Eds), Lawrence Erlbaum Associates, Mahwah, NJ.

Kolb, D. (1984), "Experiential Learning: Experience as the source of learning and development", Prentice Hall, Englewood Cliffs, NJ.

McRobb, S. R. (2002) "Draft BIS course structure and learning outcomes by level", DMU internal document, 20 September 2002.

McRobb, S. R. (2003) "BIS Course Report to Subject Authority Board", DMU internal document, 31 October 2003.

McRobb, S. R. (2004) "Notes of INFO1401 team meeting", DMU internal document, 8 September, 2004.

Mortiboys, A. (2005), "Effective and Efficient Assessment in Large Groups", staff development workshop, DMU APDU, 7 September 2005.

Prior, M. (2002) "Notes of Information Systems subject team meeting", DMU internal document, 12 April 2002.

Vygotsky, L. S. (1978) "Mind in Society: The Development of Higher Psychological Processes", Harvard University Press, Cambridge, MA.

AUTHOR BIOGRAPHY

Steve McRobb is a Senior Lecturer in Information Systems at De Montfort University, UK. He is co-author of a successful textbook on Object-Oriented Systems Analysis and Design and an associate researcher with the Centre for Computing and Social Responsibility. His research interests are in privacy online and the effect of ICT on power and trust. Steve was formerly Principal Administrative Officer at the Yorkshire Dales National Park. For more information, see http://www.cse.dmu.ac.uk/~smcrobb.

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096