In Proceedings of 2nd Australasian Conference on Computer Science Education, Melbourne, July 2-4 1997, ACM Press.

A Comparative Analysis of Design Principles for Project-based IT Courses

John C. Grundy

Department of Computer Science
University of Waikato
Private Bag 3105, Hamilton, New Zealand

jgrundy@cs.waikato.ac.nz

Abstract

Project-based courses have become increasingly popular in Information Technology (IT) curricula. We have found the design of such courses needs to take into account a variety of desired learning outcomes in order to maximise the effectiveness of such courses. This paper describes four quite different project-based courses we have developed and run over several years, and are continuing to develop. We compare and contrast the different course objectives, group management, project characteristics, course content and student assessment used in these courses. We also reflect on the evolution of these courses and how feedback from different kinds of course evaluation is used to continue their refinement. We have our experiences will be useful for others designing or refining their own project-based courses.

1 Introduction

Project-based courses have become popular in Information Technology (IT) curricula [9, 15]. This is usually due to the fostering of a “real-world” style of software development [9], the experience of working within groups on large(ish) projects [4], and the “immersion” style of learning these courses foster [3]. Such courses can be usefully utilised in many specialised IT subfields, including Software Engineering [4, 7], Information Systems [9, 12], Computer Technology and Artificial Intelligence, as well as in more introductory program design and implementation courses.

While project-based courses are seen by many as an essential component of an IT curriculum, it is often difficult to balance the wide variety of conflicting design principles that can be used when developing such courses [8, 15, 17].

For example, should such a course use industry [9] or “made up” projects [15]? Should project work be group-based [15, 10], individual [11], or a combination [17]? Should groups be self-selected [9] or organised by the lecturer [17]? How long should projects be – a whole year [10], one semester [9] or part of a semester? How should projects be managed – by students [17] or the lecturing staff [4]? Should such courses have lectures and tutorials, and if so what material should these cover [14]? Do students need preceding or subsequent “theory” courses to prepare them for or build upon their project course experiences [8, 16]? How can project work, particularly group work, be most effectively and fairly assessed – by the lecturer, by peer-review, by industry clients or a combination [9, 14]? Our experiences in project-based course development suggests these issues, and others, need careful consideration in order to achieve effective course outcomes for students.

There has been much written in recent academic literature on different aspects of project-based course design and deployment [4, 7, 15, 17]. Related topics, such as problem-based learning [1], where learning is facilitated via problem-oriented curricula, have also been widely studied for a range of disciplines. To date, there has been little analysis of the differing organisational approaches which can be used for project-based courses, particularly for IT courses. This paper attempts to address this issue by a comparative analysis of the design of four project-based, or part-project based, courses. One is a Year 3 Software Engineering project course, one a Year 3 Information Systems project course, the third a Year 4 general Computer Science project course, and the fourth a Year 2 Program Design and Implementation course. We compare and contrast the quite different natures of these courses, how the project-based aspects of the courses have been designed and refined over time, and what general course design issues need to be considered for different aspects of these courses.

2 Overall Course Objectives

The most important thing to clearly determine, and document, when designing project-based courses is the learning outcomes for the course. What kind of things should students have seen and done before they enter the course, what things should they encounter while taking the course, and what should they know and be able to do once completing it? The specific subject areas of the course need to be determined, and a course's relation to other courses, both prior and subsequent, be clearly understood. As courses with a high project-based content have heavy workloads in comparison to more traditional courses with small assignments and more lectured material, workload considerations are very important, particularly with course semesterisation [7]. Often degree requirements are important to consider during course planning, such as requirements for industry-based work experience, for which project-based courses are often ideal [16].

The four project-based courses we describe in this paper span Year 2 to Year 4 in our Computer Science Curriculum at the University of Waikato. Figure 1 illustrates the relationships between the courses. Program Design and Implementation (PDI) is a core Part 2, B semester course which all majoring students take. This contains three small individual projects carried out during the course. Software Engineering Project (SEP) is an A-semester Year 3 introductory Software Engineering course which students in the Software Engineering Programme take. Students work in groups on a single lecturer-specified project for the whole course. Information Systems Project (ISP) is a B-semester Part 3 course which students in the Information Systems Programme take. Students work in groups on an industry-based project for the duration of the course. Report of an Investigation (ROI) is a whole-year, double weighted Part 4 course which students doing Honours take. Students work individually on a single project for the whole course, under the supervision of a lecturer.

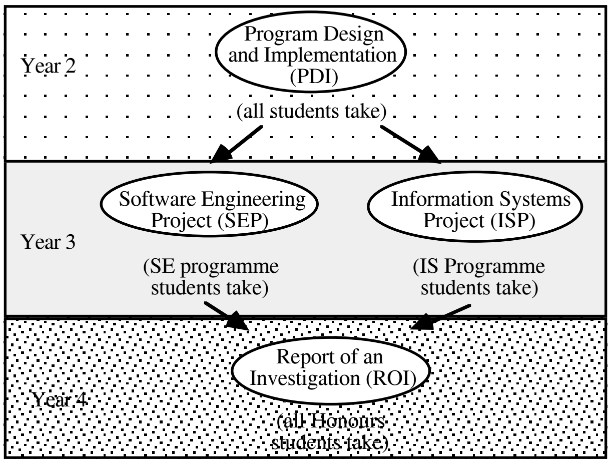
Year 2
Program Design and Implementation (PDI)
(all students take)
Year 3
Software Engineering Project (SEP)
Information Systems Project (ISP)
(SE programme students take)
(IS Programme students take)
Year 4
Report of an Investigation (ROI)
(all Honours students take)

Figure 1: The project-based courses studied.

The goal of PDI is to give Year 2 students experience in building medium-sized software systems, and follows on from a data structures programming course which contains small assignments. SEP and ISP are intended to introduce Software Engineering and Information Systems Development, with students working in groups on large projects for the duration of the course. ROI is intended to be an in-depth project on a specialised topic, chosen by the student, with individual supervision. In order to achieve these quite different course goals, we have had to design and deliver these four courses in very different ways.

3 Group Management

One of the first things to consider for a project-based course is whether the project work will be conducted individually or in groups (or whether to have some projects or parts of projects done individually and others in groups) [11, 4]. Single-person projects necessitate much more assessment (if a large course) and supervision overhead, but a single student by necessity gains an in-depth understanding of all of the project material. Single-person projects usually need to be smaller than group-based projects, and assessment issues are often simplified. However, if all students are doing the same project, unwanted collaboration may occur and be difficult to detect and avoid.

Group projects usually allow students to tackle larger, more complex problems, and give students valuable experience of working with others. Close supervision of several small groups is more realistic for lecturing staff than trying to manage many more individual projects, and there is usually less material to assess. Trying to assign differing grades to group work is very difficult [2], however, and working in a group introduces high communication and organisational overheads, reducing the actual amount of work that can be done in a course [9].

The selection of group members can be by students [9] or by lecturing staff [15, 17]. The former has the advantage that students often choose those they know they can get along with, but often group ability becomes skewed i.e. the strong students group together and weaker students group together. Lecturer-selection can result in more balanced groups, but in our experience students can react against these "enforced" groupings.

There are a variety of ways to manage project work. Recording of progress is often useful, and can be made mandatory for purposes of assessment or process improvement [17]. Project management often requires meetings of students (if group work) and possibly with the lecturer(s) (if the project work is specialised or close project supervision is needed). Project management documentation can be left to students or groups, or be a part of the assessment process. Use of project management metrics can also be used to assist students in improving their work processes, which is an important part of some project courses [17].

Group vs. Individual	• Individual • Group size: • large (4-6 people) • small (2-3 people)
Group Selection	• Self-selecting • Lecturer-selected
Project Management	• Done by student/group • Checklists provided • Meetings with lecturer • Documentation required

Figure 2: Project Organisation Approaches.

For our project courses, PDI has individually conducted and assessed projects. No explicit project management documents are provided or expected with the completed design and programming work. Students individually manage their time on the projects, which are essentially large assignments. As the course has a large number of students (approx. 150), this approach minimises project management overhead for lecturing staff while giving students larger design and programming problem experience than in our other Year 2 courses.

In contrast, SEP and ISP, work is conducted in self-selected groups on a single, large project. This approach was taken in order to "immerse" students in a single, large group project situation, with most learning thereby done experimentally. A key aim of both courses is the development of group work skills, not an aim of PDI.

In SEP groups must document their progress and hand in a variety of project management information. They also meet with the course lecturer on a weekly basis to discuss progress. The idea of Project Management, Software Processes and Process Improvement are thus covered as an integral part of project work.

In ISP, students meet frequently with their industry client, as well as with group members every week and the course lecturer every two weeks. Checklists are used to guide work, but less project management information is expected to be handed in than with SEP, with groups having responsibility for most of the project planning and organisation. This is in contrast to SEP, as in ISP the topics of Project Management and the Software Processes are not explicitly covered. Groups were originally formed by the lecturer, but most groups are now self-selected. All students have taken an introductory Year 2 Analysis and Design course in which a small group project is done, and often groups from this course carry over into ISP. Group members are thus often already experienced at how others work and how to successfully manage a project together.

In ROI, students work individually and are supervised by a member of the lecturing staff. Two progress reports and a presentation, as well as weekly meetings with the supervisor, are used to monitor progress. This approach was chosen as it allows more freedom to mange time and project tasks than in the SEP and ISP project courses, but closer supervision than the PDI course projects. Individual projects are used in ROI to allow a greater range of possible project topic choices, and to reduce the overheads of work coordination inherent in group projects, giving students more time to persue actual project work.

4 Project Characteristics

A wide variety of projects can be chosen for use in project-based courses. Industry-based courses offer the advantages of "real world" problems, a real client to discuss the project work with, and for students a satisfying feeling of solving real problems [9]. Care must be taken to ensure such projects conform to the general scope of a course, and that clients are willing and able to liaise with students or groups for the whole course. Academic-designed projects, can be more carefully tailored to cover all or most of the material the course aims to address, which can be difficult to achieve with industry-based projects [15]. Academic projects may suffer from lack of reality, however, and students feel less of an achievement in solving them.

Projects can last for an entire course and try to cover all course topics, or be somewhat smaller in size and scope. Large projects have the advantage of full (or almost full) coverage of course topics, and of giving students an "immersion" experience in the course topics [3, 4]. They also give students important experience in dealing with "big" problems, which most traditional IT courses do not [4]. Smaller, more scoped projects do not give such experiences, but can more precisely target topics of interest to the course, ensuring students gain more specialised experiences. Projects can even span more than one course (if semesterised courses are used), although this can be difficult to manage and usually requires students to take both courses in the same year [10].

Care must be taken to ensure multiple projects within a single course are not disjointed, and that workload does not become too great. Large projects, too, need to be carefully managed to ensure student workload pressures do not detract from the learning the project is intended to foster [7]. Often it is unwise to schedule two project-based courses in the same semester when a student may be expected to take both concurrently, often dramatically increasing their workload.

Project Definition	• Industry-based and defined • Academic-defined • Student-defined
Project Selection	• Self-selected • Lecturer-assigned
Project Duration	• Entire course • Part of course • Multiple projects • Spans multiple courses
Project work	• Group work • Individual work • Limited collaboration
Deliverables	• Reports • Software • Presentations • Project management materials

Figure 3: Various Project Characteristics.

Project work can be group-based or individual, and individual projects may still allow students to collaborate in limited ways. We have found it easiest in terms of assigning, managing and assessing work to make group projects contain mostly group work i.e. most work is done by all group members and assessed jointly.

Various kinds of deliverables are possible in project-based IT courses. Software and designs are usually always required, and form an important component of the deliverables, but often other written technical documents, such as Requirements documents, Evaluation and Testing reports, and comparative analyses are appropriate [6, 13]. Quality project management documents may also be important deliverables. We consider oral presentations to also be of key importance, to give students experiences in presenting their work through interactive forums and not just software or written documents [9, 13].

In our project courses, PDI uses three small projects to give students experience in designing and coding software using functional and object-oriented techniques. Projects are mostly independent, but the different approaches used are intended to be compared and contrasted by students. Deliverables are mainly software and design documents, but also include user manuals and some comparative analysis of different project results and experiences. Group projects were not included to allow students to experimentally develop skills by themselves.

SEP has a single lecturer-defined project which all groups of students study for the duration of the course. This is carefully designed so all aspects of the course material are exemplified in the project work. Deliverables are four reports and software, with all reports closely related. Two oral presentations are given by each group. Project management documents are handed in for assessment and used for process improvement analysis.

In contrast, ISP has a group-located industry-based project, and each group of students work with their different industry clients on often very different projects. Projects need to conform to a basic form, but are often quite distinct, with both small and large organisations studied. This approach was used to ensure students interact with real clients and tackle real-world problems. ISP deliverables include reports and software, and two oral presentations. Project management notes are submitted but not used in the manner of SEP. Because each group has a very different project, the content of documents and the software developed is often quite different.

We chose to have ISP students study different industry-based project and SEP students a single, lecturer-specified project because of the quite different aims of the two courses. SEP aims to provide a grounding in a range of important Software Engineering techniques and give students experiences in using these on a carefully scoped, medium-sized project. ISP is intended to give students experience in applying previously-learnt IS Development techniques on a real-world projects with real clients.

ROI has a double-weighted project which is chosen by students and supervised by individual lecturers. The main deliverable and assessment item is a report, but other items include progress reports and an oral presentation. All projects are very different in nature and focus on specialised areas of Computer Science. Individual assessment is required for this course, and as projects are so different, group work is not usually allowed. On a few occasions, when projects are similar, students have done limited work together before writing separate reports.

5 Course Content

The content of IT courses can range from very theoretical groundings, to the development of practical techniques, to building experience in applying practical techniques. Project courses usually try to build experience in applying practical techniques to a large problem, and may introduce new practices to students. For advanced projects, new theoretical insights may also be necessary to give sufficient background to the practical material.

Project-based courses often need to drawn upon theoretical and/or practical techniques developed in preceding courses, as trying to teach this material along with students carrying out a large project using it is impractical in both timing and workload [7, 16]. We have found that preceding courses which have smallish projects and some group work, along with practical material used in the project course, are an excellent grounding for students to conduct a large project [9]. Follow-on courses which further develop the experiences of project-courses are often useful. These might be more advanced project-based courses, or may delve further into the theoretical and/or practical techniques introduced in the project-based course.

Choice of software for designing and coding systems, producing documents and giving presentations can be varied. Some project-based courses may wish to make extensive use of particular CASE tools, programming environments and documentation/presentation software [9, 10]. Others may reject these tools for a more basic, and often more flexible, approach to producing project deliverables [18]. Often this decision is made not only on the availability of tools but also whether the course itself wants to foster the use of tools, review tool usage on a project or do a comparative analysis of different tools and methodologies. Often communication tools, such as E-mail, Lotus Notes and news groups can be used to facilitate group project management or discuss project-related issues.

Project-based courses usually have less of a need for lectured material than other IT courses. It may be possible to dispense with lectures altogether. Often lectures are a useful way to present examples and case-studies, or to chart expected project progress [9, 17]. If the project-based course is using techniques from another course, a brief review of these techniques through lectures is often useful [7]. If new practical techniques or theories are being introduced as part of the course, lectures are an appropriate way to introduce this material.

Focus of course content	• Theoretical issues • Practical techniques • Experience in using techniques
Preceding and subsequent courses	• Cover theory/practice used • Build upon project experiences
Software choice	• CASE tools • Programming languages/tools • Documentation/report writing • Presentation tools • Communication tools
Lectures etc.	• Lectures on theory/practice • Tutorials for case studies, tools • Labs for tool usage • Meetings with students/groups
Course materials	• Textbook(s) • Lecture notes • Manuals (for tools) • Case study examples • Project management materials

Figure 4: Project-based Course Content.

Lectures and meetings with students or student groups can be complemented with tutorials and/or laboratory sessions. As much of the assessment and deliverables of the project-based course are practical development work, keeping tutorial and laboratory requirements to a minimum is often necessary to keep student workload to acceptable levels. Tutorials and labs are a possible alternative to lectures for introducing practical techniques or presenting case studies or examples of tool usage.

Additional course materials, ranging from text books, case study documents and systems, and on-line resources can be used to complement lecturing. Often project-based courses can most usefully utilise textbooks or notes focusing on case studies or practical examples, rather than theoretical issues. Recommended and/or required readings and various on-line or library resources are useful to provide background information for course topics.

For our project courses, PDI has three lectures a week plus a tutorial for all students. Lectures cover both theoretical and practical material, and the project work for the course is intended to give students experience with applying the practical techniques on well-scoped example problems. Laboratories are staffed by demonstrators who assist students with tool usage. Particular programming and documentation tools are used (C++, Gofer, Netscape and MS Word), as well as a News reader and course-specific news group. A programmers textbook for C++ and a Gofer manual are used. Example programs and a course manual are provided on-line via the World Wide Web.

SEP has two lectures per week, which focuses on mainly Software Engineering theory and practice material, in addition to weekly meetings of groups with the course lecturer. No tutorials or laboratories are used, and students are expected to reuse many practical techniques taught in PDI and other Year 2 courses. Timesheets are used to monitor student workload. Students are given freedom to choose particular CASE tools, programming tools and documentation tools. Part of the course content is a comparative analysis of tools, and discussion of experiences in using different tools on the project. Considerable new Software Engineering theory and practical techniques are introduced, including Requirements Engineering, Specification, Software Architectures, Implementation choices, Testing and Maintenance. Lecture material covering these topics is complemented by a textbook focusing on Software Engineering theory and practice, and a variety of on-line resources are used. Course material is distributed via the World Wide Web. Lectured material is kept to a minimum with students expected to make use of these other resources.

ISP has two lectures per week and meetings with the course lecturer and demonstrator on alternate weeks. ISP uses practical techniques and tools taught in the preceding Year 2 Analysis and Design course. Particular CASE tools (Deft), programming tools (MS Access), communication tools (E-mail) and documentation tools (MS Works) are used extensively. ISP introduces limited new theory and practical techniques in the areas of Information Systems Planning and Project Management. A textbook focusing on Information Systems Development practice is used to complement lectured material. All lectures focus on a case-study problem of a fictitious ISP group going through a project. This was found to be much more successful and useful for students than abstracted theoretical and practical Software Development methodology examples (as used in SEP).

ROI has one lecture per week for the whole year, used to chart expected progress for students. Students also take turns at giving an oral presentation during this time. As all ROI projects are so different, no material is taught in lectures, but students are expected to do all work and manage their project with the individual supervisor's assistance. This approach provides some guidance to students, and gives students an opportunity to review other students' project progress. The course coordinator is also an experienced member of staff who can give high-level support and encouragement for this individual project work.

6 Student and Project Assessment

Assessing project-based courses, particularly the assessment of group work, is recognised as one of the major potential difficulties of project-based courses [2].

Group work can be assessed equally for all group members, with all members receiving the same grade for their work [9, 15]. This is usually the easiest way to assess group work, but can result in unfair grades for members who have done better work or contributed more to the project than others. Attempting to take into account the amount and quality of work on a group project can be very difficult. It is almost essential to require detailed project management information from groups if this is to be done [17, 18].

In most project-based courses, the lecturer is responsible for most or all of the assessment. Peer assessment and assessment by an industry client can also be incorporated.

Peer assessment of project work allows other students to comment on their colleagues work. Peer assessment can be incorporated into a project-based course in several ways. Presentations allow for comment on others' work, as do demonstrations of software. Peer assessment may involve a student or group assessing another group's work, or may involve a student assessing the work of other students in the same group. All are valuable sources of feedback from other students. Some care needs to be taken that peer assessment is reasonable and fair, particularly if peer assessment is an important component of overall course assessment.

Group work	• same grade for group work • items of individual assessment • use project management notes
Individual Assessment	• Tests • Examinations • Essays • Individual reports, parts of reports
Assessors	• Lecturer/tutor/demonstrator • Peers in other groups • Peers in same group • Industry clients
Process vs. Product	• Product = quality of deliverables • Process = quality of how done (based on project management etc.)

Figure 5: Assessment Strategies.

Individual assessment of project work can be valuable for distinguishing between individuals in the same group, where work is demonstrably done by one person. Tests and examinations are common individual assessment techniques, but other approaches include individually produced essays, reports or parts of reports. Project management information can be used to isolate aspects of group work performed by one individual for purposes of assessment. If this is done, care must be taken to ensure the project management information is correct i.e. all group members agree it is an accurate reflection of work on the project. We have found that requiring all group members to sign such documents helps to ensure group agreement on its accuracy.

There is some debate about the assessment of process i.e. how project work was done vs. product i.e. the end products of the work [17]. It is generally acknowledged that assessing process is quite difficult, but as the process needs to be repeated on different projects, quality of the process can give a better reflection of student capability than product [17, 18].

In our example courses, PDI requires three small projects and a final exam, with all work being individually carried out and assessed by the lecturer. While this has proved to be a good approach, with 30% of assessment from the exam and 70% from project work, we are exploring the possibility of making one of the projects group-based, to give PDI students an introduction to working in groups.

SEP is internally assessed with four deliverables (reports and software), two oral presentations, a test and an individually assessed component. All deliverables and presentations are group-assessed i.e. all group members receive the same grade. The project management notes, meetings with the lecturer and peer assessments are used to determine the individual grade.

ISP is internally assessed with a final report and software, test, two oral presentations and individually assessed grade. Unlike SEP reports, the final report has earlier feedback on drafts from both industry client, demonstrator and lecturer. Peer assessment of presentations (groups on other groups) and individuals (group member on other group members). Industry clients are asked to provide feedback on project quality, the manner in which students conducted the project work with the client, and suggestions for course improvement. The lecturer determines the individually assessed grade using peer assessments, client feedback and notes from group meetings. Unlike SEP, the project management information is not used as extensively and the final product quality is weighted more highly than process.

ROI has the bulk of the assessment based on the final report. The oral presentation is peer-assessed as well as being assessed by members of the lecturing staff. Interim reports are used as guidance of project progress, not for assessment. This approach is used because of the very disparate nature of the projects in this course.

7 Course Evaluation and Evolution

Like other courses, project-based courses need to be constantly evaluated and improved, to ensure learning outcomes are being achieved, and that delivery of material and project-based experiences are suitable for desired outcomes.

Evaluation of courses can be done by several people. The lecturer needs to be proactive in reviewing courses and in ensuring the projects chosen, project management, course content and assessment approach achieve the desired learning outcomes for the course. Students must be given adequate opportunity to comment on the projects they have done and suggest improvements in course content, delivery and management. Feedback from past students who are now in industry is often difficult to obtain, but we have found this is very valuable in validating course quality and various design principles used. If a project is industry-based, obtaining feedback from industry clients on the student's progress and course content is important.

Evaluation by colleagues in other institutions and by professional bodies may also be appropriate [16]. If a course or degree programme needs to conform to recognised standards, then critical appraisal by these external people is necessary to ensure continued suitability of project-based courses.

Carried out by	• Course lecturer • Students • Other staff • Industry clients • Colleagues from other institutions • Professional organisations
Methods of assessment	• Student evaluations • Client evaluations • Comparative analysis to previous years • Workload assessment • Comparison to other courses and other Institutions' courses

Figure 6: Course Evaluation Approaches.

We have found a careful comparison of results to previous years is often a valuable exercise. Project deliverables, test results, individual assessments and project management information can all be usefully compared [15].

Improvements to project-based courses may be made as a result of feedback from students, industry clients, other staff, comparison to previous years' results or other external factors. We have found student feedback to be the most valuable in our experience. A number of project-based courses which have had external pressures to reduce workload, particularly from other staff, have required dramatic modifications, often resulting in splitting courses up or drastically modifying their content and/or nature [7].

In our project courses, PDI uses student feedback in the form of evaluations of the course and review by other staff, as it is a core Year 2 course. We have evolved this course over the past two years to change the programming languages used, to introduce different design methodologies, and to revise the projects required. We used to require four large projects as the entire course assessment, but have reduced the project size and split the first project into two smaller assignments. This has reduced student workload and allowed us to provide a better flow on from the first semester data structures course that precedes PDI.

SEP has evolved considerably from a small individual assignment-based, theory course to a practice and experience group project-based course. This was in response to both staff and student feedback, and in examining trends at other Australasian Universities. Our Software Engineering curriculum was lacking in a project-based course to give students experience in using theoretical ideas and practical techniques on a group project, which SEP now provides.

ISP has evolved from lecturer-selected groups to self-selected groups, and a range of case study and project management materials have been introduced. Industry clients and projects have always been used, but the student projects are now more carefully managed with closer project guidance from lecturing staff. These changes were in response to industry client and student feedback, and by comparison of results to previous years. More recent comparison of results seems to justify these changes, with improved projects, project management and student evaluation feedback.

ROI has been modified to require attendance at the weekly lecture/presentation, to require progress reports and to give staff better guidelines for project report assessment over the past couple of years. Previously the course had been run in a very decentralised way and the comparative quality of project work and reports was unacceptable. Comparative results from 1995 and 1996 indicate the changes to this course have been very successful. This course has undergone regular evaluation by colleagues at another New Zealand University to ensure quality of outcomes and assessment.

8 Summary

It is generally recognised that project-based IT courses contribute significantly to students' understanding of applying theoretical ideas and practical techniques to large problems. Group projects also foster valuable development of cooperative work skills, and the documentation and presentations required in project-based courses develop often-neglected written and oral skills [5, 13]. Designing project-based courses needs to take into account a range of factors, however, in order to achieve desired learning outcomes for students in quite disparate courses which use the project approach. Such considerations include the overall goals of the courses, if group projects are used and how projects are managed, whether projects are industry-based or academic-designed, what supporting lecturers, tutorials, tools and materials are used, how assessment is carried out, and how the courses themselves are evolved in response to various kinds of evaluation.

We are continuing the development of the four example courses described in this paper, but also applying the design principles outlined above to other courses we offer or plan to offer. The Year 2 Systems Analysis and Design course project is being reviewed to determine possible changes to improve learning of SA&D and basic system implementation. A Year 3 Graphics and Multi-media project course is being designed which will utilise individual G&MM projects under lecturer supervision. A Year 4 Human-Computer Interaction course being run this year is using a group project approach to give students experience in applying HCI theory and practical techniques on a large project. We are also investigating the possibility of using small group projects in Year 1 introductory courses, to show students that group project work is an integral part of IT system development.

References

[1] D. Boud and G. Feletti. *Challenge of problem based learning*. London: Kogan Page, 1991.

[2] S. Brown and P. Knight. *Assessing Learners in Higher Education*. London: Kogan Page, 1994.

[3] N. Chrucher and A. Cockburn. An Immersion Model for Software Engineering Projects. In *Second Australasian Conference on Computer Science Education*, Melbourne, July 2-4, 1997.

[4] R.F. Cohen and T. Menzies. Providing Software Engineering Students with an Experience in 'Big Computing'. In *Proceedings of the 1994 Software Engineering Education Conference,* M. Purvis Ed, pages 71-77, IEEE CS Press, Dunedin, New Zealand, November 21-24 1994.

[5] S.J. Cunningham. Learning to Write and Writing to Learn: Integrating Communication Skills into the Computing Curriculum. In *Proceedings of the 1994 Software Engineering Education Conference,* M. Purvis Ed, pages 306-312, IEEE CS Press, Dunedin, New Zealand, November 21-24 1994.

[6] R.A. Day. *How to write and publish a scientific paper*. Oryx Press, fourth edition, 1992.

[7] G. Dobbie and G. Bartfai. Teaching Software Engineering in a Computer Science Department. In *1996 Software Engineering: Education and Practice Conference*, pages 58-63, IEEE CS Press, Dunedin, New Zealand, January 24-27 1996.

[8] G. Ford. *1990 SEI Report on Undergraduate Software Engineering Education.* Technical Report SEI-90-TR-3, Software Engineering Institute, Carnegie-Mellon University, 1994.

[9] J.C. Grundy. Experiences with Facilitating Student Learning in a Group Information Systems Project Course. In *1996 Software Engineering: Education and Practice Conference*, pages 12-19, IEEE CS Press, Dunedin, New Zealand, January 24-27 1996.

[10] S. Hope and J.E. Terry. A Recursive Student Project to Reinforce the Principles of Software Engineering. In *1996 Software Engineering: Education and Practice Conference*, pages 68-75, IEEE CS Press, Dunedin, New Zealand, January 24-27 1996.

[11] E.A. Kemp. The role of the individual project in teaching Knowledg Acquisition. In *1996 Software Engineering: Education and Practice Conference*, pages 138-143, IEEE CS Press, Dunedin, New Zealand, January 24-27 1996.

[12] S.E. Little and D.B. Margetson. A project-based approach to Information Systems design for undergraduates. *Australian Computer Journal*, Volume 21, Number 2, 1989.

[13] J.G. Meinke. Augmenting a Software Engineering project course with oral and wiritten communications. *SIGCSE Bulletin*, Volume 19, Number 1, pages 238-243, January 1987.

[14] M.J. Oudshoorn and K.J. Maciunas. Experience with a Project-based Approach to Teaching Software Engineering. In *Proceedings of the 1994 Software Engineering Education Conference*, pages 220-225, M. Purvis, IEEE CS Press, Dunedin, New Zealand, November 21-24 1994.

[15] M.J. Oudshoorn, A.L. Brown and K.J. Maciunas. Simulating Real-Life Software Engineering Situations in the Classroom. In *1996 Software Engineering: Education and Practice Conference*, pages 20-25, IEEE CS Press, Dunedin, New Zealand, January 24-27 1996.

[16] G. Roy and V. Verarrt. Software Engineering Education: from an Engineering Perspective. In *1996 Software Engineering: Education and Practice Conference*, pages 256-262, IEEE CS Press, Dunedin, New Zealand, January 24-27 1996.

[17] V. Veraart and S. Wright. Software Engineering Education – Adding Process to Projects: Theory, Practice and Experience. In *Proceedings of the 2nd Asia-Pacfic Software Engineering Conference (APSEC'95)*, pages 148-157, IEEE CS Press, Bisbane, Australia, December 6-9 1995.

[18] V. Veraart and S. Wright. Experience with a Process-driven Approach to Software Engineering. In *1996 Software Engineering: Education and Practice Conference*, pages 406-413, IEEE CS Press, Dunedin, New Zealand, January 24-27 1996.