CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Federation ResearchOnline

Journal of Information Technology Education Volume 6, 2007

A Scalable and Portable Structure for Conducting Successful Year-long Undergraduate Software Team Projects

Kathleen Keogh
The University of Ballarat, Ballarat, Australia
k.keogh@ballarat.edu.au

Leon Sterling
The University of Melbourne, Melbourne, Australia
leonss@unimelb.edu.au

Anne Venables
Victoria University, Melbourne, Australia
anne.venables@vu.edu.au

Executive Summary

Year-long team projects with external clients provide a well recognized opportunity for students to gain industry experience, whilst being supported and guided by staff to minimize risks. Each group should be supervised to ensure that they have enough direction and confidence to approach a new problem of significant size, without being daunted. A structure is needed that is flexible and adaptable to suit various institutional cultures but, at the same time, provides the safety net to ensure that success is likely. This paper presents a reflective analysis of teaching at three different institutions and presents the resulting distilled wisdom of experience that has produced a structured framework for capstone project units.

The proposed structure is scalable to any class size and portable across institutions and potentially across technical disciplines. The structure leads to team student projects that are successfully engaging and provide excellent experience toward producing work-ready graduates. The structure is flexible in design so that the teaching workload does not increase too much as class sizes increase, but students are still well supported with appropriate scaffolding and mentoring.

We detail the key factors in our framework: careful project selection, appropriate sign posts, and helpful guides that together improve upon overall project success. We argue that students need appropriate projects, 'good' clients, and a well formed team. The sign posts support students through the 'uncharted waters' of ill-defined problem-based learning. Sign posts include: deliverables, deadlines, team notebook, progress reports, and accountable time-tracking. Helpful guides spread the teaching workload broadly: relying on cultural expectations, technology, team supervisor, feedback, and clear marking guidelines.

This structure has been successfully implemented in at least three different universities in Australia and several remote partner teaching sites in different countries over many years. We have provided case study examples of implementations in three different Australian Uni-

Material published as part of this publication, either on-line or in print, is copyrighted by the Informing Science Institute. Permission to make digital or paper copy of part or all of these works for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage AND that copies 1) bear this notice in full and 2) give the full citation on the first page. It is permissible to abstract these works so long as credit is given. To copy in all other cases or to republish or to post on a server or to redistribute to lists requires specific permission and payment of a fee. Contact Publisher@InformingScience.org to request redistribution permission.

Editor: Grandon Gill

versities: a large urban university (The University of Melbourne), a small urban university (Victoria University) and a regional university (The University of Ballarat).

Students benefit from such an experience in their final stages of an undergraduate computing degree. The scaffolding and support provided by the structure enables students to improve their skills in technical development, communication, team work, project management and client negotiation. The project unit provides opportunities for independent self directed learning and realistic experiences in an environment that allows for experiential learning. Students are engaged when working with real problems for real industry clients. Our student feedback supports our view that students' confidence in their ability to manage a project is enhanced through their project experience. Based on our own observations and anecdotal feedback, we believe that our students benefit and are helped to become 'industry ready' graduates through the learning experienced in the capstone project course.

This same structure could be used successfully in similar programs. We would encourage others to consider adapting this framework to their institutional context, as we have collectively found it a positive model to help us manage the considerable workload in offering successful year long undergraduate software projects with clients.

Keywords: Undergraduate Capstone Projects, Software Engineering Education, Course Design

Introduction

Producing industry ready graduates who are able to perform successfully in the workplace is a very important goal for tertiary educators. Factors of importance to employers, as described by Nunan (1999), extend beyond technical capabilities and include flexibility, independent learning, highly developed communication skills, and the ability to work in teams. This is underlined by industry accreditation requiring teamwork opportunities be provided during the undergraduate experience. Organising teamwork so that it is fairly structured and assessed is not an insignificant challenge (Clarke, 2002, 2005), as students should be sufficiently engaged to ensure that participation is equitable between team members.

Engagement is an important issue more broadly. Engaged students become more deeply involved in their own learning, and the current generation of students are very strategic with their choices for engagement (Krause, 2006). Students are highly motivated when faced with authentic tasks in a realistic setting (Krause, 2006).

Year-long team projects with external clients provide a well recognized opportunity for students to gain industry experience, whilst being supported and guided by staff to minimize risks. Each group needs to be supervised to ensure that they have enough direction and confidence to approach a new problem of significant size, without being daunted. A structure is needed that is flexible and adaptable to suit various institutional cultures but, at the same time, provides the safety net to ensure that success is highly likely. This paper proposes a structure that is scalable to any class size and portable across institutions and potentially across technical disciplines. The structure leads to team student projects that are successfully engaging and provide excellent experience toward producing work-ready graduates. The structure has been successfully implemented in at least three different universities in Australia and several remote partner sites in different countries over many years.

This paper describes and reports on the positive aspects of capstone project units at three different institutions. The work presented is valuable as a presentation of our considered experience and observations of how to manage projects in a flexible framework that we recommend to others teaching similar units. This paper is a comprehensive review of distilled wisdom – based on literature and our own experience. The paper is organised as follows. We initially provide some

background regarding our approach and argue that it is based on current best practice. We articulate key success factors for student software team projects, concentrating on those related to student support. We outline the structure in terms of the scaffolding it provides for problem based team learning with a constructivist approach. We then show that it is flexible and can be implemented in different settings by providing case study examples and short descriptions of its successful implementation at various institutions.

We conclude this section with some comments on terminology. A project can occur within a Software Engineering degree, IT degree, or Computer Science degree. The different degree context can give subtly different scope for a project, and student background and expectation can differ. The project is often described as a capstone experience. People differ between whether the project is done by a team or a group. Discussing the trade-offs in terminology is beyond the scope of the paper. What we have in mind is a year-long software development project for an external client by a team of at least four later-year students, where students have to proceed from requirements elicitation through to delivery of a product to the client.

Background

Software team projects are recognized by professional societies, computing practitioners, and educationalists as a core component of effective undergraduate computing degrees (Fincher & Petre, 1998). When offered as a capstone course, such projects provide motivation and opportunities for students to consolidate their understandings of "systems analysis, software development lifecycles, specific software design support tools, entity relationship modelling, entity life histories, database design, web site design, or web server programming" (Daniels, Faulkner & Newman, 2002, p. 121). More importantly, when students engage in problem-based and experiential learning they become active participants in their own learning, constructing their own internal knowledge through experiences of the social and physical context of their work (Brooks & Brooks, 1999; Dart, Johnston & Schmidt, 1996; Greening, Kay, Kingston, & Crawford, 1997; Kolb, 1984; Kolb & Kolb, 2005; Lynch, Heinze, & Scott, 2007; Newman, Daniels & Faulkner, 2003; Tam, 2000).

It has been well reported that software team projects also help students develop more generic skills, such as problem-solving, and allow them to assume responsibility and improve their communication abilities. By exposing students to collaborative work and team building, projects have been associated with changes in student attitudes and an appreciation of industry ethics and professional perspectives (Clarke, 2005; Lynch, Goold, & Blain, 2004; McLay, Corich, & Millma, 2005; Newman et al., 2003). Partially in recognition of these benefits, many institutions pursue work-based projects, whether authentic or in simulated learning settings (Bennett, Dunne, & Carre, 1999), as a mechanism to improve their overall graduate outcomes.

Yet despite all of the documented benefits, software team projects are not without their problems. Although undergraduates report valuing the experience where they recognize the importance of teamwork and collaborative learning (Crebert, Bates, Bell, Patrick, & Cragnolini, 2004), they can find adapting to such (relatively) large scale projects somewhat disconcerting. Software engineering capstone courses are "pedagogically different from the standard courses within their program" (Lynch et al., 2004, p. 432). Newman et al. (2003, p. 95) comments on a "noticeable mismatch between the conventional academic approach, where achievement is measured as individual success with a focus on demonstrating technical knowledge, and the building of successful practical computer based systems which requires collaborative team working and is highly people oriented". Fleming (2005) reported upon problems created by differences between student and staff expectations in a New Zealand software engineering subject. Fleming describes students as adopting one of two coping strategies, neither of which was desired by the staff. The first resulted in students doing lots of extra work and the second was dropping out of the subject en-

tirely! Similar experiences are reported by Conn (2004, p. 496) who warns that for some students in software team projects "Stress is a killer, and some students showed signs of extreme levels of stress at times".

For academics the importance of a software team project in final year computing degrees is without argument (Daniels et al., 2002; Fleming, 2005; Gehrke et al., 2002 Lynch et al., 2004; Newman et al., 2003). However conducting such crucial courses in a Computer Science (CS), Software Engineering (SE) or Information Technology (IT) degree is expensive, labor intensive, time consuming, complex, and continually demanding (Fincher & Petre, 1998). Pedagogical models should inform delivery mode decisions (Lynch et al., 2004) in light of student preferences and their needs for support (Clear, Young, Goldweber, Leidig, & Scott, 2001; Martin, 1997). The challenge, as pointed out by Gehrke et al. (2002), is in attempting to maximize the practical experience for each student whilst exposing them to project complexity. At the same time, the academic motivates students to complete the appropriate documentation, enforce software engineering standards, encourage best practice, and ensure the final delivery of high quality software.

To answer the challenge, we argue, from the literature and our experiences gathered in three disparate tertiary institutions, that a scalable and portable organizational structure is needed to facilitate student learning in team projects, particularly where student groups work on separate software developments. Transfer of good practice between institutions and cultures does not happen without some modification and adaptation (Fincher, Petre, & Clark, 2001; Lynch et al., 2007). The structure we advocate is not rigid. Rather we name the key elements of support that, if present, will promote an engaging and successful project experience. We recognise that students should be well supervised as well as supported. The characteristic of our structure is in the provision of three key elements: careful project selection, appropriate sign posts, and helpful guides that together improve upon overall project success.

A Scalable and Portable Structure

When undergraduates commence their software team subject, they are expected to work on different projects, each with separate demands generating unique challenges. Projects may cover diverse technical areas, including software development on specialised platforms, hardware design, networking problems, and multimedia development. The students may possess some of the technical knowledge but fall short on experience in the application of software engineering techniques (Dawson, 2000; Gehrke et al., 2002). Some of our students have completed previous study units in project management and have familiarity with project management tools; however this experience has been limited to tasks such as the creation of gantt charts for assignments, rather than using these as live applications in a project of 11 to 24 weeks duration. Based on our student surveys, students often begin their project lacking in confidence and have not previously had opportunities to experience estimation, work allocation, and team collaboration of the scale required in their project. Although often highly motivated to achieve success in delivering a working system to their client's satisfaction, in reality students often do not have the management, planning, or team coordination skills to see a project through to completion. They often make mistakes along the way. They are in 'unfamiliar territory' and the constructivist teaching philosophy (Tam, 2000) dictates that students should be allowed to learn from experience (Newman et al., 2003). Instructional scaffolding as described by Linder, Abbot, and Fromberger (2006) can provide social or academic support to computing students, enabling them to engage with problems that would otherwise be out of reach.

Whilst the project experience drives their learning, students should be supported adequately to empower them to develop their skills and adjust their understanding of issues. Yet, our experience has shown us that it is not helpful if students are taught in a prescriptive manner. Rather, learning is best supported by the presence of good 'sign posts' and guides. Positive experiences

are predicated on suitable selection of appropriate projects and good clients, who are supportive of the educational aims of the project. Together, these help provide safeguards to ensure that projects are more likely to survive and reach maturity, even when problems are encountered.

Table1: Key factors in our proposed structure linked with literature

The Structure	Corresponding Literature	Description/emphasis
A. Project Selection		
1. Appropriate projects	Goold (2003)	Characterisation of appropriate projects
	Lynch, Heinze & Scott. (2007)	Obtaining suitable projects identified as a major challenge in both the Australian and South African case study
	Kribar (2005)	Unique topics
	Fleming (2005, p. 94)	Realistic topics designed to push students and they should be "not quite doable"
	Daniels, Faulkner & Newman (2002)	Viability of task should be the first thing considered then size of project
2. "Good" clients	Daniels, Faulkner & Newman (2002)	Need understanding from real world clients that they might not get their problem solved
	Gehlke et al., (2002)	Staff should not act as clients, students should deal with real customers
3. Team formation	Daniels, Faulkner, & Newman (2002)	A group project does not need all members to have the same skills
B. Sign Posts		
1. Deliverables	Goold (2003)	Specification of project deliverables
	Kribar (2005)	Regular reports and documentation
	Lynch, Goold, & Blum (2001)	Define top level deliverables
2. (Loose) deadlines	Kribar (2005)	Structured deadlines
	Gehlke et al., (2002)	Deadlines are needed for students
3. Team notebook or repository		
4. Team progress	Goold (2003)	Provision for reflection, analysis and review
5. Budgeted hours, time tracking, time sheets	Lynch, Heinze, & Scott. (2007)	Time management is identified as a main challenge in the South African case study
C. Helpful Guides		
1. Cultural Expectations a. Learn the necessary technology	Goold (2003)	Adequate support of capstone courses with prior curriculum
2. Team Supervisor		
3. Mentoring and Training Supervisors	Goold (2003)	Effective people management
4. Early Document Feedback	Nicol & Macfarlane-Dick (2006)	Importance of feedback supporting formative assessment
5. Marking Guidelines	Goold (2003)	Transparent assessment Clear definition of goals
		Reports that students want clearer guidelines and marking schemes

Good sign posts provide guidance and direction along a path and their value should not be underestimated. In reviewing the literature to identify best practice, Goold (2003) identifies several sign posts, clear definitions of goals, and the specification of deliverables as being most important. Others echo their importance (Fleming, 2005; Hribar, 2005; Mann & Smith, 2005).

We have mapped some of these factors identified in the literature to the corresponding elements of our structure in Table 1.

Our structure includes signposts that provide an additional, albeit flexible, structure to support students in the uncertain and sometimes uncomfortable experience of problem-based learning. We have found that our signposts provide students with some guidance, security, and confidence for students in their progression through year long software team projects.

Project Selection

Selection 1. Appropriate Projects

The choice of projects given to students has an impact on the student experience. Authentic and realistic problems are very engaging and, hence, motivating. Designing problems for students to encounter can shape the student experience (see for example: Dawson, 2000; Fleming, 2005, Gehrke et al., 2002). We agree with Garlan, Gluch and Tomayko (1997) that it is important that the student project is not crucial to the client, so students, whilst motivated to produce a product for a real client, are free to learn by making mistakes. Projects should be viable and appropriate in size (Daniels et al., 2002). In our experience, if a project is too small the team is not motivated to engage in appropriate design and analysis methodology. It is not an insurmountable problem if a project has to be re-scoped during the project. Ideally, projects are selected that are scalable up or down depending on the abilities, skills, and experiences of the team. Together with the supervisor, the group should scope the project and, in liaison with their client, prioritise the requirements that will be designed and implemented in the final solution. The subject coordinator and the teaching staff who have project experience provide an additional reservoir of experience and knowledge.

It is important that particular resources needed for each project are available. If particular resource requirements are otherwise unavailable to students, the client provides them. The technical strength of the degree is maintained with a cross section of projects that require technical and design skills. The students should be able to learn new technologies for their project, however it is also good to match groups' skills and interests with the project they are assigned.

Based on our experience, students can benefit from a project designed to extend and deepen the technical skills developed in prerequisite units or, alternatively, students can be challenged to independently learn new technical skills. The scope achievable in the project duration should be balanced with the amount of self-learning required to achieve a solution. Developing the confidence to engage in independent learning of a new technology is a desirable graduate quality (Nunan, 1999).

The project, whilst important to be flexible and scalable, need not be technically too challenging in order to be a valuable learning opportunity for students. In our experience, difficulties that arise are generally due to problems with team management or communication rather than technical problems. As students generally enrol in capstone projects in their final year, their technical skills are well developed according to their chosen specialty. Project supervisors find that the students need significant mentoring to develop an understanding of different personality styles and how to work on a substantial project in a collaborative way. Estimation and allocation of work in an equitable way to team members with the appropriate skills is sometimes difficult; for example,

when some team members do not have good written English skills, then the burden of proof reading and corrections to documentation may rest with one or two team members. In many cases, the capstone project is the first time that students are involved in group work that is truly collaborative – in the sense that in order to complete the project, tasks must be divided between the team, but not done without significant commitment to regular communication. As is discussed in the next section, signposts are helpful in directing the students and providing an initial process framework for students to follow.

We concur with Dawson (2000) who strongly advocates that learning through realistic experience with real clients and ill structured problems rather than well defined and carefully planned projects is more beneficial and has been successful over an extended period of time. In a recent survey conducted at The University of Ballarat, 19 out of 20 students answered yes to the statement "It has been great dealing with a real client". Nine out of the twenty students also replied yes as to whether they had experienced difficulties working with a real client.

Selection 2: "Good" Clients

The main goal, from the students' perspective, is to succeed in delivery of a system that they have scoped, researched, designed, and implemented for their client. Through the process of eliciting clients' requirements, students are given the opportunity to learn and understand the importance of correct analysis. The focus at this level is on correctness with respect to client needs and usability. From the beginning, clients are made aware that our educational aims focus on the process rather than product and these aims are achieved during the project experience (including making mistakes!). At worst, clients are guaranteed to receive good documentation describing their requirements, and it is likely they will receive a working system that implements at least some of the core requirements. The most important qualities desired of a client are supportiveness and willingness not to interfere with the students' processes. Clients still behave as typical clients and many clients are initially vague and change their minds about requirements. Ideally, the requirements specification is signed off towards the end of semester 1. If there are significant client difficulties, it is possible for a staff member to resolve these (even if this means taking over as pseudo client) without jeopardising project success.

The task of sourcing clients/projects is a significant overhead; however, once a database of projects and clients is established and clients experience the engagement with students, it is possible to establish relationship with clients who are prepared to propose further projects in subsequent years. At the University of Melbourne, where this process has been in place for over ten years, we have clients who have returned to submit new projects, and our process has matured to the level where we now confidently charge an administrative fee from clients as a way of funding the infrastructure we need to support the project teams.

Ideally, the staff coordinator can source projects prior to the start of semester so that students can focus on team formation and project selection early in the semester. We have tried allowing students to find their own projects based on their family/community contacts; sometimes this has been very successful, particularly when a student has a direct family relationship with their client. It can introduce problems and significant delays with starting the project if students do not have access to potential clients. Our strong preference is to establish a process where by potential clients can submit their project proposals a few weeks prior to semester starting. We have found it productive to send letters to local community groups, schools, and business inviting proposals. We have found it useful to set up online application process so that clients can propose their projects easily online. This enables storage of the project proposals in a central location and enables staff follow up later. Where the institution has a relationship with industry partners, it can be good to engage corporate clients. (If students do have family connections with a small business, we have directed them to invite their potential client to propose a project independently).

When project proposals are received, investment of time by the staff coordinator to talk to each client and discuss the project aims and structure is time well spent. This can be a good time to identify clients or projects that may not suit the student project. Problems have arisen when we have had clients who wanted to interfere with the processes we are asking students to follow or if clients are not in the position to be flexible regarding timelines or documentation produced. If a client suggests projects that are low risk and not essential, students can create a system that is useful either as a proof of concept or as a working prototype. In some cases, clients have chosen to pay one or two project students to do further work on their prototype as part of vacation employment.

Selection 3: Team Formation

In order to create a team that will work well together, some time spent initially on team formation is valuable. It is important that each team is balanced with respect to a diversity of skills (e.g. technical, communication, documentation, organisational) and share similar expectations regarding effort and final grade. Following team selection and identification of team interests /project interests, the next requirement demanded from our students is to decide upon their team and management structure. Roles and responsibilities as well as team standards and processes should to be negotiated and documented to avoid later problems. Our experience has confirmed that encouraging students to consider their project management and team management as evolving as the team processes establish and mature is valuable. We enforce this by requiring that students update their formal management documentation throughout their project experience. (Sign post 1 in the next section refers to documentation deliverables.)

There are pros and cons with letting teams choose themselves. The advantages are that students often prefer to work with their friends and, when confronted by a new learning experience, are reassured by not having to negotiate new social relationships. It is usually less work on the subject coordinator. The cons are that teams are unbalanced and less likely to be diverse. In our experience, diverse teams work better due to the need to be clearer about communication and a broader range of skills that can be brought to bear on the problem.

Matching student teams and projects is not a simple process and it requires some judgement. Daniels et al. (2002) detail the two basic options: students are assigned to projects or students chose their projects. One idea we have successfully used is to engage students from initial stages by inviting groups to place ‘tenders’ for projects that interested them. Based on the strength of each tender, projects were assigned to groups. Alternatively, groups placed bids for projects and staff allocated projects based on a judgement of students’ abilities and skills looking at their previous grades only. In the tendering process, students were invited to argue more explicitly their appropriateness and to highlight the team expertise and skills more directly. Processes that encourage students to articulate their skills, attitudes, and capabilities are to be encouraged.

Diversity in a team is important and, at the start of the project, we have found it valuable to invite students to reflect on their abilities and be open with their team regarding how they can best contribute. A team without a natural organiser who can fulfil the role of project manager can lead to disaster. In at least one case, one author reflects upon, a team failed to produce a successful project solution for their client due to lack of organisation. In that case, the team had strong programming skills but did not appreciate the need to manage the project from a broader perspective, so ended up with a solution that did not address the client’s needs carefully enough. Their design approach was limited, and they overlooked any significant consideration of the user interface (which was significant as theirs was an educational spelling tool for young children 6-12 years old). Often, we find that students initially underestimate the effort and time that will be required to meet with, report to, and consult their client. The rewards of adopting good processes of management, good recording of minutes, and formal team meeting procedures are not initially recog-

nised unless students are explicitly motivated to adopt some processes. As is discussed later, the assessment structure can be used to motivate students to value process and not just focus on their final product in isolation. Developing these skills and learning through experience that process is helpful in team work is valuable toward producing graduates that are not only skilled with technical programming skills, but more generic skills such as time management, communication and team collaboration.

Appropriate Sign Posts

We have developed the following 'sign posts' to provide guidance to our project students. Importantly, they are not meant to be highly prescriptive or to detail how a task has to be achieved. They are intended to guide the student experience toward success and minimise some of the risks inherent in the student project experience. These signposts provide sufficient scaffolding to support the students without prescribing and directing their performance too much.

Sign Post 1. Deliverables

In many disciplines, a project of significant size could be described and documented in terms of a set of requirements to be met by the project, a design for the solution, a plan for verification, and validation of that solution. We propose that students be required to articulate such documentation in a formal way – to help ensure that client requirements are well understood and to ensure that design decisions are thoughtfully considered and evaluated. For success, it is essential that all team members know their roles and responsibilities and the team rules and standards that are to govern the team processes. These process standards can be described in a project management plan (PMP) and all team members can sign their agreement to these as a contract between all team members at the beginning of the project.

In our experience, we have required students of SE and IT to produce the following documents, based on the IEEE standards: Software Project Management Plan (SPMP), Software Requirements Specification (SRS), Design Document (SDD), and Test Plan (TP). The project management plan is very important to the development of the student team and features in the curriculum of project based courses elsewhere (See for example Garlan et al., 1997). In our experience with inexperienced undergraduates, the early negotiation of roles and responsibilities is initially a very theoretic exercise and students have difficulty in grasping the importance of this document, however anecdotal feedback from students on final project completion has supported our perceptions of its importance. Students have said, "I wish I realised the importance of the PMP initially, it was the most important document we wrote". Having clearly stated roles and responsibilities helps the team develop collaborative work habits and is important to developing an appreciation of the various team roles (Purvis, Purvis, & Cranefield, 2004). We encourage teams to allocate roles appropriate to their project including, for example: Project Manager, Secretary, Librarian, Analyst, Graphic Designer, Client Liaison, Quality Assurance Manager, Designer, Test Manager, and Programmer.

In some years, we have had students submit a Software Quality Assurance Plan (SQAP) rather than the SPMP. However we have found that students in third year do not have enough ability to appropriately document their quality assurance procedures. We rely heavily upon the quality assurance provided by staff mentors or other students in their roles as auditors and external reviewers. Students are also advised to produce a Risk Management Plan. During the project, students are encouraged to think of these as "live" documents and to maintain versions of their documents using versioning tools made for this purpose (eg. SCCS, CVS). The final version of documentation is assessed with an expectation that improvements have been made subsequent to initial feedback. As will be discussed in a later section, guides provide timely feedback to students as

the documents are being produced. The final grade apportioned to deliverables is also based on the final system delivered, including appropriate user documentation.

Sign Post 2. (Loose) Deadlines

To aid project planning on a macro level, students are given a set of major deadlines for some significant milestones in the project, such as delivery of documents: SPMP, SRS, SDD and TP. On given dates, each team provides evidence of progress. This evidence may be by the submission of an electronic version of the document(s) to be delivered and/or by the presentation of an oral report. This evidence is not necessarily directly graded for assessment. However students receive timely feedback directly from some of their guides relating to their work (discussed in a later section). These deadlines, by their presence, provide a framework for project planning and encourage students to work on appropriate tasks in the software development lifecycle. The teams are told to establish and work to their own project plans. The date for the team's completion or base lining of each of their documents does not necessarily coincide directly with the deadlines formally imposed.

Sign Post 3. Team Notebook

To encourage students to develop good record keeping skills, each team is also asked to maintain and submit a team "notebook" on completion of the project. It is expected that team decisions and design notes will be recorded here, including evidence and reports of any related investigations undertaken. Such reports may include feasibility studies, technological investigations, and research reports. This notebook is open for public scrutiny and we encourage our students to 'publish' their notebook electronically on a project webpage. This transparency allows for coordinating staff and student peers at any stage to review team progress and provides a window into the processes followed by each team.

Sign Post 4. Team Progress

Toward the end of the first semester, when teams are finalizing their requirements, obtaining a sign-off of these with their client, and beginning to work on their architectural design, each team is asked to present an oral progress report to the class. We have conducted these as formal 15 minute presentations and as less formal poster sessions. These sessions have proved to be valuable, both in providing feedback and as a conduit toward students maintaining momentum, thereby encouraging the development of useful prototypes and providing an opportunity for technical communication and justification of design decisions made thus far. Such sessions are also valuable in providing teams with background to other teams' projects and technical issues. They allow teams to hear how other groups are organizing themselves and the management processes that have been adopted. Students can informally rate themselves against other groups to judge how well they are performing and to note management or organisational processes that others are using that might be worthy of consideration.

In addition to the oral reports made to the entire class, we have found that when multiple staff are involved as supervisors, a half hour team interview with the principal staff coordinator to discuss team progress and any problems or issues of concern is useful to students. It also allows coordinating staff to review the progress of each team and to provide some individual feedback to students about their team processes and experience thus far. At the first interview, the focus was on team formation and project management and initial progress. For the second interview, students were asked to report on their initial testing and provide some evidence of their early test results. If appropriate, progress relative to other teams can be discussed. A final interview may also be used as an assessment tool to allow individual students a chance to demonstrate some of what they have learned in their own reflections of the team experience.

These signposts are very successful at leading students and providing a level of guidance and direction. This ensures that, with appropriate individual attention and feedback, student teams will follow the correct processes to ensure quality and increase the likelihood of success in their project. The individual team attention is provided by several people as discussed in the following section.

Sign Post 5. Budgeted Hours, Time Tracking

To support project management and fair allocation of work amongst team members, we have tried a number of different strategies. It is clear that students should be accountable for their own contribution, and if there are non-participating students there is objective evidence gathered to ensure that remedial action or (in extreme cases) individual assessment of students occur rather than a joint team mark. Elsewhere strategies for fair assessment of team projects are discussed and we do not focus on this aspect here. (See for example: Clark, Davies, Skeers, 2005; Clarke, 2005) We do however note that there is a need for some guidance to ensure that students with little prior management experience can successfully complete their project.

One successful strategy has been to make students directly accountable for their time. Each team was asked to keep timesheets or similar records using management and task tracking tools. Students were given a 'budget' of hours available for each group member (10-15 hours per week) and if they 'overspend' their budgeted hours without substantial justification, this will be reflected in their grade for project management skills. Students were given the incentive to manage their workload on a team basis. An alternative approach we've tried is to ask students to manage their time internally and to give their supervisor a regular team report, including individual time sheets, but not impose a strict hourly budget system.

Helpful Guides

Professional wisdom suggests that students have the opportunity to practise management and experience group dynamics beyond theoretical exercises so that they can think independently and adapt to realistic problems likely to be encountered in the workplace (Falkenberg, Russell, Ricker, 2000). It is therefore important for future competency that students create good conceptual models of the tasks involved in Software Engineering and Project Management. Fincher and colleagues (2001) describe transfer of ideas and materials as successful when ownership occurred with that transfer and when accompanied by adaptation and change in the adoption process. In addition to the sign posts provided to lead students toward a good model of Software Engineering practice, it is important that the students are mentored and guided to adapt processes to suit their individual project needs. This guidance need not be provided by a single individual; in fact it can be enhanced if several different people provide guidance. The following guides are suggested:

Guide 1. Cultural Expectations.

Students learn from their peers. In our experience, project units develop a reputation amongst students. The reputation can be helpful if students have access to previous work examples that set the standard expected. It is valuable for them to be able to consult peers who have already completed or are currently completing a similar project unit. Over time, we have come to rely more heavily on guidance and support that has grown among our student body. Advice and information is passed on, formally and informally, from both graduates to final year students and from fourth year students to third year students. Educationally, there are a number of goals held for the students. It is hoped that students understand, through their experience, the value of structure and formality in their communication and the importance of planning and management.

Peer interaction can be expanded into a formal mentoring experience. Indeed we have come to expect that mentoring skills are included in the professional skills developed by students during their course. To give a specific example, the 4th year students at the University of Melbourne, having recently learnt by their own experiences in their 3rd year project, are invaluable as peer guides for the current 3rd year students. As a component of the assessment of 4th year, these students are asked to fulfill the following support roles for a 3rd year team:

- External process auditor;
- External document reviewer;
- Provide advice based on their own experience during focused interactions at weekly workshops on related topics.

Each 3rd year team is given 4 audits and reports on their processes during each phase of their project life cycle as negotiated between themselves and their 4th year auditors. In addition, each of their major documents is externally reviewed by a 4th year student at least once.

At the University of The University of Ballarat, we have observed cultural change amongst the project unit teams since 2004 when the project curriculum was updated. In 2005 students clearly benefited from the past documents from 2004 provided as examples, though students were still uncomfortable with the ill-defined tasks. In 2006, students appear more settled and when surveyed at the end of semester 1, showed they clearly valued the process and documentation they were required to work on. Informally, students were encouraged to discuss and review documentation for other teams. From 2005, students were required to individually formally review a piece of documentation from another team. This peer review has proven to be very productive and a positive experience for the reviewer and the team being reviewed.

Guide 1a. Learn the Necessary Technology

Each project generally brings with it unique challenges to students in terms of learning new technology. Projects present an opportunity for students to take responsibility for their own learning in a new programming language or technical skill. In some cases, students have been heard to say that they chose a particular project in a new area so that they had the opportunity to learn some skills that otherwise they wouldn't have covered in their degree. Other students choose to undertake projects in areas where they already have expertise, to add value to their CV for future job searching. That students willingly engage in learning new technology without formal instruction or support is part of the cultural expectations that the project subject brings. There is value in the students developing confidence in informal self-learning of new skills beyond a structured teaching environment, an opportunity that the students appreciate provided they know that this is expected.

Guide 2. Team Supervisor

As Daniels et al. (2002) discuss, close supervision of groups by staff is best. Each team is allocated one supervisor, who is expected to spend an average of 5 hours a week on team supervision and related duties. The supervisor "travels" with the team for the duration of the project. Some of our best experiences at the University of Melbourne have been with supervisors who are recent graduates of the Software Engineering program who have completed 3rd and 4th year projects. We have also used staff as supervisors.

The supervisor may not necessarily have the technical ability to advise on team problems but provides social support on teamwork issues. Managing group conflict is a significant part of the supervision role as student workloads can become a problem if a group is not managed well (Fleming, 2005). Garlan et al. (1997) also discuss the value of the mentor in their studio projects to support, guide, and encourage innovation with postgraduate students; in our case, with under-

graduate students we find that the team supervisor can provide this mentoring both at the team and individual level as required.

The entire team or team member taking the role of project manager schedules a weekly meeting with their supervisor who provides the team with mentoring and guidance. The supervisor keeps abreast of individual issues faced by the team and only if necessary, involves the coordinating staff to resolve problems. The supervisor is akin to a high level manager, interested but not a part of the team directly. Duties of the supervisor may also include facilitating weekly software engineering student workshops attended by 3rd and 4th year students to discuss topics and share their experiences.

Guide 3. Mentoring and Training Supervisors

Supervision of our undergraduate projects is aimed toward developing competent and confident software engineers. To provide support for the supervisors and some consistency of experience between teams, the supervisors meet with coordinating staff on a regular basis to share experiences, plan workshops and moderate between teams. Other regular communication occurs via email providing collegial support amongst supervisors in setting workshop agendas and resolving issues that arise during the project. Regular feedback between the supervisors is important toward the aim of providing consistent supervision to all teams as much as possible.

Guide 4. Early Document Feedback

The students at all institutions are required to submit versions of documentation at key milestone dates during their project. The supervisors review the documentation and give detailed comment. These comments form invaluable feedback to assure students that they are 'on track' with their project. In our experience, this feedback may or may not be associated with formative assessment tasks. In some cases, allocating marks to early versions of documentation helps motivate students. We have found this motivation especially helpful when there is not a strong existing culture surrounding the project units. Students at the University of Melbourne are not given formal marks until the end of the year, whilst students at the University of Ballarat and Victoria University are given progressive assessment marks based on each submission, with additional marks awarded on management, process, and final versions at the end of each semester. Feedback to students is very important to support formative assessment and self-regulated learning (Nicol & Macfarlane-Dick, 2006).

Guide 5. Marking Guidelines.

It is a common perception that students are motivated by the final grade they receive for a subject. To capitalise on this the students are given access to the marking guidelines used for the final assessment. These make explicit the percentages allocated for each aspect of the project. In this way, students are motivated to work toward improving their *processes* to maximise their mark. Students want clear guidelines and marking schemes (Fleming 2005). The marking scheme can therefore encourage students to focus on developing their professional work-ready skills in communication and project management and not be tempted to focus only on the final product. The general breakdown of marks for projects, modified slightly to suit each institution is:

Deliverables (25%) comprised of

- Software Requirements Specification 10%
- Final Release 15%

Process (75%) comprised of

- Project Management and Quality Assurance 25%
- Analysis and Design 25%
- Implementation and Delivery 25%

The focus is on process and it is expected that students have improved and developed this for the duration of the project. Artefacts that are examined in the assessment of process include documents such as SPMP, Risk Management Plan, Design Notebook, SDD, and Test Plan. In addition, the team repository, team minutes, and project plan are inspected to audit whether or not the SPMP processes have been followed as documented.

Implementation

In this section, we provide specific examples of assessment and case study descriptions of specific projects to help the reader appreciate the scale and value of this structure. In the subsequent section, we profile three institutions that have successfully implemented this model. A typical time line for a 2 semester project would be as follows:

Semester One:

Week 1-3: Team formation, Tender bid creating and project selection

Week 4-7: Further team formation, writing PMP, first client meeting

Week 8-12: Further client meetings, initial analysis of requirements, preliminary design and creation of first executable prototype.

Semester Two:

Week 1-4: Finalising details of design

Week 5-9: Programming and system building, planning for testing

Week 10-12: Testing, writing user docs, finalizing all documentation

Assessment

The assessment scheme used for the capstone project needs to be adapted to the students and take into consideration the student profile and particular aims for each institution. Table 2 shows three assessment models for projects typical of each location. At The University of Ballarat and Victoria University, the project units are assessed at the end each semester, however at the University of Melbourne, the assessment is based on the entire year as one unit. Shading differentiates tasks as either process tasks or product output related tasks. The overall approach is similar for each institution; however there are some notable differences. For example, at the University of Melbourne, with a rich culture and strong peer support in the student projects within the Software Engineering degree, the assessment does not rely as much on assigning marks to drafts during the project, rather on final assessment of documents (with an expectation that there are a number of revisions and updates during the project). When introducing this model for projects at the University of Ballarat in 2004, with no real historical support from strong process or project management in previous years' projects, the assessment tasks strongly emphasised the allocation of process marks for client management, team collaboration, and project management. This was helpful to highlight the importance of these aspects of the project.

Table 1: Key factors in our proposed structure linked with literature

The Structure	Corresponding Literature	Description/emphasis
A. Project Selection		
1. Appropriate projects	Goold (2003)	Characterization of appropriate projects
	Lynch, Heinze & Scott. (2007)	Obtaining suitable projects identified as a major challenge in both the Australian and South African case study
	Hribar (2005)	Unique topics
	Fleming (2005, p. 94)	Realistic topics designed to push students and they should be "not quite doable"
	Daniels, Faulkner & Newman (2002)	Viability of task should be the first thing considered then size of project
2. "Good" clients	Daniels, Faulkner & Newman (2002)	Need understanding from real world clients that they might not get their problem solved
	Geluke et al., (2002)	Staff should not act as clients, students should deal with real customers
3. Team formation	Daniels, Faulkner, & Newman (2002)	A group project does not need all members to have the same skills
B. Sign Posts		
1. Deliverables	Goold (2003)	Specification of project deliverables
	Hribar (2005)	Regular reports and documentation
	Lynch, Goold, & Blain (2004)	Define top level deliverables
2. (Loose) deadlines	Hribar (2005)	Structured deadlines
	Geluke et al., (2002)	Deadlines are needed for students
3. Team notebook or repository		
4. Team progress	Goold (2003)	Provision for reflection, analysis and review
5. Budgeted hours, time tracking, time sheets	Lynch, Heinze, & Scott. (2007)	Time management is identified as a main challenge in the South African case study
C. Helpful Guides		
1. Cultural Expectations a. Learn the necessary technology	Goold (2003)	Adequate support of capstone courses with prior curriculum
2. Team Supervisor		
3. Mentoring and Training Supervisors	Goold (2003)	Effective people management
4. Early Document Feedback	Nicol & Macfarlane-Dick (2006)	Importance of feedback supporting formative assessment
5. Marking Guidelines	Goold (2003)	Transparent assessment Clear definition of goals
		Reports that students want clearer guidelines and marking schemes

Assessment tasks motivate students to value process; initially perhaps for marks, but eventually we hope for value. It is difficult to examine the direct educational merit gained by students in their project experience, but we do know that students experience satisfaction at completing high quality material for a real client. Our student feedback provides us with informal data on which to reflect upon the value of our model. Generally, feedback from students includes comments: "lots of work involved, but great when we finish and see what we have done". As well, students who

engage with their project achieve excellent final grades along with an industrial strength experience. A recent graduate commented, "The project units were awesome; they gave you a real life experience in all facets of project management and the phases involved. Our project was very relevant to future employment and the teamwork aspect enhanced our team working skills in readiness for industry employment" (Feedback during University of Ballarat, Courses Review, BIT (Prof Prac), 2007). This supports the anecdotal feedback from students returning for graduation ceremonies and confirms that students value their project experience as a good preparation for their future career work. On completion of project unit, a student commented, "I am definitely more confident in my ability as a Project Manager and it has confirmed that this is my chosen career. I think it will also look really good in my resume and I really hope the client was happy with our efforts" (Comment by student, July, 2007).

Typical Staff Experience

As we have outlined, our structure provides support to students from a variety of sources, so that the role of supervision is not entirely left to one staff member. From our experience, staff supervisors are often busy and overloaded, and in some cases, the project supervision role is an 'extra' load. In this section, we outline typical experiences and suggested responsibilities of the coordinator and supervisors.

The staff coordinator of the capstone project spends time at the beginning of the project establishing the potential project list and interviewing and meeting with potential clients. It is best to establish good processes for sending out the letters to potential clients, keeping records of client details and notes on each potential project. On occasions, projects have not been accepted if the coordinator feels that the client will not be able to support the educational aims of the project. For example, if the client has strict timelines or demands for a particular process that doesn't fit with our methodology.

Another significant task for the coordinator prior to project initiation is creating infrastructure support for project teams. For example, electronic repositories can be created by establishing 'team room' space in content management systems such as Blackboard/WEBCT (http://www.blackboard.com) or by allocating server space for a website for each team. It is usual to also create chat rooms, newsgroups, blogs, or equivalent opportunities for inter-team communication and announcements from staff. In addition, scheduling lectures to provide background information at appropriate times during the semester is helpful, not only as an opportunity for imparting information, but also to generate a collegial atmosphere between groups. We have found that teams can get competitive, but when students meet regularly in class for lectures and for facilitated discussion workshops, the peer support is significant.

The lecture topics chosen depend on the background of the students; the lectures can be specific on topics such as project management, software engineering – analysis and design techniques, validation and verification, and quality assurance. We have found that students respond well and particularly value lectures that are less technical in focus, covering 'common sense' topics such as team formation, how to conduct your first client meeting, managing the client relationship, and reviewing documentation (describing purpose and examining example documents).

Especially during the first semester, less formal workshops have also been very successful. During workshops, students talk with students from different teams and compare experiences, such as how was the first client meeting, what roles, responsibilities and rules teams have adopted; and compare (and sometimes review) documents between teams. The value of these workshops became obvious when such workshops were optional and majority of students chose to attend. On more than one occasion, the coordinator at the University of Ballarat, semester one in 2007, would leave the workshop and students would remain in the room continuing with their discus-

sions. One student was quoted as saying to their supervisor 'I can afford to skip lectures in other units, but not in project as I need to hear what you are going to say' (K. Keogh, personal communication, June, 2007).

The team supervisors may be less engaged with the project experience due to their workload, so meeting regularly (every 2-3 weeks) is important. Such meetings can include discussion of style of supervision, marking of documents, and any specific problems. If supervisors appear reluctant to embrace project management process or are inexperienced, these meetings can be used to attempt to ensure consistency in the support provided by supervisors. The coordinator can also gauge if interference is needed. For example, when a team was experiencing difficulties with communication, one approach we used was to announce (to all teams) that the coordinator wanted to be invited to attend a team meeting within the following fortnight. Then the coordinator could appear to 'discover' by probing questions to the team that further action was needed.

Supervision style does differ between people, as one author reflects some supervisors interfere and can almost manage their team to success (on one occasion, one author learnt of a supervisor who even did some coding to help a team that was in trouble), whilst perhaps the preferred model is to guide and suggest but not enforce good practice. We believe a significant value in the project experience is that the students learn through the experience – including by making mistakes. The typical supervisor experience would be to hold a weekly 1 hour meeting with their team during semester one and then less frequently (every two to three weeks) during semester two. If the teams have well established processes, meetings can be replaced by email status reports on some occasions. The supervisor would also be expected to read and review draft documentation during the project.

The mentoring and support provided by supervisors to help with team conflict, particularly how to deal with a non-contributing student or an apparently 'lazy' team member, is very important. The first author reflects on a situation when a team (and supervisor) decided to kick a non performing student out of the team rather early into the project experience. The student was then isolated from the team but allocated a 'contractor role' to produce independent (but not critical) work toward the final product. This led to continued problems for that student who did not complete his work and ultimately failed in the project. It is difficult to know if that student's level of engagement could have improved with different management. In the same semester, a different supervisor, with a team in a similar situation, implored the team to be tolerant and keep records of contribution and evidence (e.g. emails, minutes) of attempts to engage a non-performing team member. Progressively, that student improved and completed enough work to pass - with a lower grade than their team mates, but with significant learning (across the entire team) regarding the need for explicit communication regarding team work.

The supervisor plays a significant role in subjective assessment of the team and their adopted processes. When the final assessment is made on a team, particularly with respect to process, evidence is examined and the supervisor reports on their own impressions regarding the team cohesiveness and management. The workload of the project supervisor would be at least equivalent to taking a weekly tutorial or lab class in a more traditional unit.

Success Stories

We present three very different case studies of project experiences to help the reader understand the diverse experiences that can be involved for students and clients.

Case study 1

Often prospective clients for student projects have been planning an upgrade of their commercial website. This was the case in 2003 when the federal executive of the Australian Licensed Air-

craft Engineers Association (ALAEA) approached Victoria University with the idea of allowing students to update their website. The ALAEA organisation (ALAEA, 2007) which represented the industrial, technical, and professional interests of Licenced Aircraft Maintenance Engineers in Australia had been frustrated by the inability of their existing website to respond sufficiently quickly to issues of concern for their members, such as occurred during the financial collapse of Ansett Airlines in 2001.

Early in the academic year, five students met with their client and elicited the user requirements. A dynamic and interactive web system offering three-tier access for ALAEA members, ALAEA staff members, and the public was planned. The proposed system would have the added functionalities of login, a content management system, membership update capability, mass email system, online voting system ,and online custom discussion forums. By mid year, the Systems Requirements Specification had been done and signed off by the ALAEA executive.

Then disaster struck. The student group splintered over personality clashes and concerns that individual members were not pulling their weight. As a result, academic staff spent a great deal of time supporting and counselling each team fraction attempting reconciliation. Finally, with the consent of the client, the decision was made to allow each splinter group to independently see the project through to completion. For the client, the outcome was the delivery of different two systems from which they choose one for full implementation. Subsequently, the following year the ALAEA executive privately contracted one of the students to continue fine tuning the installation and further develop the website, which is now the public face of the organisation (ALAEA, 2007). As a result of the project experience, the client has acted as a professional referee for each participating student; some securing very good graduate positions in industry and government as a consequence.

Case study 2

At the University of Ballarat, the students involved in projects can have quite distinct interests and skills including networking, multimedia, databases, and gaming. (The next section outlines the different degrees that students may be enrolled in.) A small cohort of high achieving students are engaged in a industry-based course, in which they study along with periods of part and full time professional practise work with IBM, in partnership with the University of Ballarat. In 2006, a team of such students was asked to work with the Security Team in IBM™ Australia to develop a prototype to show the potential for developing a children's game. The intent of the game was to educate 8-9 year old children about safe interaction with internet activities including appropriate and safe behaviour in chat rooms, when internet browsing, instant messaging ,and appropriate recognition of spam/junk emails.

The team of 5 students did not have particular expertise in graphic design or gaming, however they embraced the challenge with enthusiasm. The client was not particular regarding requirements for the content of the game; the game script, characters, and features were left to the imagination of the students. The client did have strict requirements regarding the security awareness features that should be covered and obvious requirements that the system should appeal to primary aged children. The student team interacted with a client team that was split over multiple geographic locations (Sydney, Melbourne, and Ballarat). This remote collaboration worked very well as the client team was experienced with this, working this way on a regular basis. The students held regular meetings with status reports to clients and supervisors. In their first semester, the students developed the game script and storyboard. The game had a superhero theme – various heroes had come from another planet to ensure that Earth children were safe online. The feedback to children playing in the virtual game world was provided by these superheroes. The game was developed using Flash and an initial prototype was created in semester one. More detailed character profiles, graphical design, and the GUI were developed in semester two. The sys-

tem design and bulk of the programming was completed before technical and user testing. The students arranged for real children at targeted ages to play with their system. Feedback on this was collected to inform their design.

The students did have some technical difficulties in establishing how best to integrate their flash animations with a database. They used an external database to store their scenario choices and also to save a history of achievements for each player so that play could be resumed in a later session. The quality of the graphics was good enough for the proof of concept prototype. As the team members were not experts in this area, they made a conscious decision not to attempt high grade graphical animation; however the final product looked very good and was found to appeal to the target audience. The delivered system included a teacher disc (CD) and a student CD. The teacher disc provided features to enable tracking of player progress with the game. Finally, the system was delivered in a very professional way, complete with colourful packaging and CD labels.

Feedback from IBM Security was very positive in terms of the benefit of the prototype the students produced, and they plan to extend this concept further into a more complete game. In the next stage of this development IBM in Ballarat recruited the project students upon graduation with the object of utilising the same 'small cohort' to develop the game further.

Case study 3

Projects at the University of Melbourne have come from a variety of sources, including industrial partners, government agencies such as the Bureau of Meteorology, and from colleagues in other parts of the University with software to develop. The software can range from language teaching tools, through databases, image processing, video annotation, multimedia, and visualisation. The case study to be described had an unusual client, the Teaching and Learning Quality Assurance Committee of the University. It was interesting in its scope and also spawned a short-lived startup company.

The project was to build a tool for detecting plagiarism, at a time where there was growing concern about plagiarism in the media. Note that one of the early deliverables for the project was a survey of existing plagiarism detection tools. The survey was well received and was circulated within the university and influenced the content of the university's Academic Honesty Web site.

Clearly electronic access was needed to the student assignments in order for software to compare them. The student team suggested an electronic submission system as phase one of the project. The change of scope was accepted. The students talked to several stakeholders across the university to understand the diversity that needed to be accommodated within the submission system. While essays were clearly a mainstay, they also needed to consider computer programs, spreadsheets, and images.

Additional requirements, which proved useful, included distribution of workload and formation of teams. For the former, essays could be batched to individual tutors to mark, with tutors able to access the papers they needed to mark. For the latter, students could self-elect to teams, which had proved a logistic difficulty without the software. A feature added later was a translation of the interface to Norwegian, due to a visiting Norwegian student who joined the team for the second half of the year.

The project was successful, with a prototype delivered by the end of the year. The Requirements document was particularly well done. It has served as an example in subsequent years.

Two of the students were particularly interested in extending the project. They stayed on after their final year, improving the system, and running a trial on plagiarism detection within the University in the first semester of the following year. Eight subjects, involving almost a thousand

students, were chosen for the trial from five different faculties and included a large first year Law subject, an Arts subjects, an architecture subject, and later year computing subjects. The trial ran successfully and showed only a modest amount of plagiarism. The software was demonstrated at the opening of the ICT Building at the University of Melbourne as an example of student projects.

The university made it clear that they would only adopt the software if it came from an external supplier rather than from inside the university. The two students asked some colleagues from the software industry to form a company: Queue Solutions. A modest amount of funding was raised from running a couple of small projects and from some university funding to prepare a bid for ongoing service provision. Several presentations were made to the university. Ultimately turnitin.com (iParadigms, 2007) was chosen by the university for its plagiarism detection tool, and the company petered out.

Institutional Experiences

In this section, we profile three institutions that teach year long projects using the structure outlined in this paper. We highlight their similarities and differences to support our argument for the scalability and portability of our structure across disciplines and technologies and with students of diverse backgrounds (cultural and educational). All three institutions teach project units successfully according to the structure described in this paper. The University of Melbourne is a large urban university with high numbers of students from diverse backgrounds on campus. Project students are all studying 4 year engineering degrees majoring in Software Engineering. Victoria University is a smaller urban university, also teaching project to students off shore in Hong Kong and Malaysia; all students doing the project are completing 3 year Science degrees, majoring in Computer Science, Mathematics, Information Technology, or Internet Technologies/Applications. The University of Ballarat is a smaller regional university with growing international student numbers on campus, but the majority of on campus students are Australian born. Significant cohorts of The University of Ballarat international students are enrolled in campuses elsewhere in Australia (Adelaide, Geelong, Melbourne, Sydney) and off shore (Hong Kong, New Zealand). The University of Ballarat project students are studying degrees in Information Technology, Information Systems, or Applied Computing with streams in Mathematics and Games technologies.

Year long software team project subjects have been running at the University of Melbourne for over 15 years. Third year students work in teams of four or five to develop a project for a client. Initially, students came for several different courses, a three year B.Sc. degree, majoring in Computer Science, a four year Software Engineering degree, often as part of a combined degree, or later, a three year Bachelor of Computer Science Degree. Discussion of aspects of running the software project subject at the University of Melbourne subject during the first several years can be found in Dart, Johnston, Schmidt, and Sonenberg (1997).

As student numbers boomed in the late 1990's, the third year software project subject was restricted to students undertaking a Software Engineering degree. The Software Engineering students undertake an additional year long project subject in their fourth year. The fourth year project team is typically larger than in third year, ranging from 10 to 15 students. The scope is correspondingly larger. Interaction between third year and fourth year students has been valuable, and has been reported in Dart et al. (1996).

Over the fifteen years, there have been at least eight subject coordinators, with several tutors, and over one hundred supervisors. While there has been changing of the material provided, workshops conducted, and fine-tuning of assessment and deliverables, the overall feeling of the subject

has been relatively consistent. Indeed trying to explain the ability of the subject to remain constant despite a large variation in subject coordinators was the initial inspiration for this paper.

The projects in 2006 included 3-D argument visualization, a contacts network site, accounting software, electronic boardroom, corporate governance management, managing social clubs, personal time manger, maps on CD, middleware for integrating office software, automated messaging software, and a web-based abalone reporting system. Previous years had similar diversity.

At Victoria University all students are enrolled in a Bachelor of Science in the following streams: Computer Science, Computer Science and Mathematics, Information Technology, Internet Technologies and Applications, must undertake at least one year long undergraduate software team project, known as the industry project, in the third and final year of their course. Additionally, students have a mandatory enrolment in a co-requisite English language unit of study that focuses on providing support for the documentation of the industry project. This language and presentation support is particularly important since the majority of students are non-native English speakers at this University. The number of students enrolled per year in the project unit across all campuses has ranged from 20 to over 200 during the past decade. Regardless of the student numbers, the same structure has been successfully used.

For each campus location, a staff supervisor is responsible for the organizational aspects of all group projects. Their task involves initial liaison with real clients, sourcing and organizing the available projects, overseeing groups in their formation, assigning fellow staff to act as individual group supervisors, setting deadlines, timetabling of presentations, and the collation of results. Each group of 4 students has an academic supervisor with whom they meet with on a weekly basis. The supervisor offers project specific advice and discusses management problems with their own groups. In addition, these supervising staff act as panel members for 2 hours at 2 to 3 separate timetabled group presentations a semester. At the end of semester, supervisors meet with the coordinator for a workshop to grade and rank all the projects and the individual grades of each student in the industry project subjects. For the staff, a supervision of one project group is considered as the equivalent of a teaching hour.

In the co-requisite English language subject, weekly classes cover a diverse range of topics, like software documentation, job applications, and resume preparation. Sessions are conducted by language and communication lecturers. These academics assist students in preparing their project documentation by detailing correct layout of documents, such as Project Descriptions, SRS, and Test Plans. By assisting project students in their drafts, the workload is reduced for computing staff assessing the same documents for content. Types of projects include creating specific middleware for clients to integrate legacy software, tailored content management systems for Websites, VoIP projects, and eye movement capture software.

At The University of Ballarat, students enrolled in a year long project can be studying a number of different courses: Bachelor of Applied Computing (Mathematics or Gaming), Bachelor of Information Systems, and Bachelor of Information Technology. In some courses, project is a compulsory unit, in others it is chosen as an elective stream. In the Bachelor of Applied Computing (Mathematics), it is compulsory to take the first semester of project, but not necessarily the second semester. Students doing a Bachelor of Information Systems only do the first semester of project. The first semester is concerned with team formation, requirements gathering, design, and preliminary prototype building; the second semester is concerned with detailed design, building, quality assurance, verification, validation, and user documentation. Our teams are comprised of a mixture of students from any of the courses; students self select their team and together bid for a project that interests the group. The total number of students enrolled in the project unit at a location has ranged across our campuses from one or two teams (less than 10 students in total) to as many as 150 students.

One staff member is allocated to coordinate the project units at each location. Every team is allocated a supervisor for the duration of a semester; generally a team has one supervisor for the duration of their project, but sometimes in the second semester, the supervisor might change. Supervision is spread across full time teaching staff with varying backgrounds – in mathematics, computer science and information systems. There is one unit coordinator who delivers support lectures and coordinates workshop discussions on a regular basis. The unit coordinator provides support to the supervisors. Supervisors meet regularly to discuss team progress and moderate marking. Each supervisor marks the artifacts produced by their team, but marks are moderated in a group moderation session with all supervisors. Marks are given to students throughout the year as artifacts are produced; marks are also awarded for process (including document improvement), management and teamwork.

Our off campus delivery is presented by lecturing staff employed by partner teaching institutions who are provided with the assessment requirements, marking guidelines, and lecture materials. The unit coordinator at the home institution keeps in contact with the lecturers at partner institutions and sometimes attends student presentations. All material from partner sites is moderated by the University of Ballarat staff to ensure consistent quality is maintained between locations. Generally the staffing model off campus is quite different, with the lecturer delivering all lectures, running discussion sessions, and supervising all teams. However, the assessment and teaching material is identical across all campuses.

Prerequisite units in project management and software engineering analysis must be completed prior to the project unit. In addition, students in streams for multimedia, games development, or networking will have completed background studies in these areas prior to beginning their project. Projects undertaken include software development projects, networking projects, game development, multimedia DVD projects, and electronic commerce/website projects.

Our projects have been a great tool for community engagement, and our clients are sourced from local community, schools, small business, and corporate clients. Teams are encouraged to bid for projects of interest and with clients close to home, however for various reasons we can end up with teams based in Sydney working on projects with remote clients in different geographical locations. Our teams show the adaptability of the structure we use as the style of projects and technology used are very diverse but can all share a common framework with success. We also have postgraduate coursework students who do a scaled down one semester project in their final semester of study.

The structure presented in this paper was introduced at the University of Ballarat in 2004 to replace a very loose existing structure in which the entire project experience was based on the individual supervisor and no guidelines existed for deliverables, milestones or process. With the informal structure, the University of Ballarat had produced a number of successful student projects, however with the introduction of the new structure in 2004, the overall quality and professionalism of all our project students increased significantly. Feedback from staff, students and clients has been very positive.

All students, regardless of their location, are required to have an electronic team repository where all their team documents and management material, meeting minutes, and other records are kept. Teams are encouraged to look at the work of current and past teams available on team web sites. This interaction across locations has helped us to create a new culture within projects and students' expectations of the project experience. In any semester, we have quite a variety of projects, and the unit webpage has links to all current project teams, at every location. There is also inter-team interaction in a more formal way with individuals performing external reviews for another project team.

Conclusion

Over the last decade we have gained valuable insights from our involvement with teaching and supervising undergraduate Software Engineering student projects at our respective universities. This distilled experience, supported by the literature, has helped us formulate a successful, scalable, and portable structure for year long undergraduate team projects. The structure is flexible and adaptable to suit diversity in project types, student background, team sizes, and staffing structures with real clients. We have detailed the key factors in our framework, namely: project selection, signposts, and helpful guides, and have outlined example implementations in three different tertiary institutions.

Students benefit from such an experience in their final stages of an undergraduate computing degree. The scaffolding and support provided by the structure enables students to improve their skills in technical development, communication, teamwork, project management, and client negotiation. The project unit provides opportunities for independent self-directed learning and realistic experiences in an environment that allows for experiential learning. Students are engaged when working with real problems for real industry clients. Our student feedback supports our view that students' confidence in their ability to manage a project is enhanced through their project experience.

A goal for educators is to produce industry ready graduates who are able to cope with a diverse range of problems and apply theoretic knowledge in a practical way. Our structure provides students with the opportunity to learn constructively with a realistic experience. Appropriate support is provided in several ways to minimise the workload associated with supervision.

This same structure could be used successfully in similar programs. We would encourage others to consider adapting this framework to their institutional context, as we have collectively found it a positive model to help us manage the considerable workload in offering successful year long undergraduate software projects with clients.

References

ALAEA. (2007). Australian Licensed Aircraft Engineers Association home page. Retrieved 26 September 2007, from http://www.alaea.asn.au/

Bennett, N., Dunne, E., & Carre, C. (1999). Patterns of core and generic skill provision in higher education. *Higher Education, 37*, 71-93.

Brooks, M. G., & Brooks, J. (1999). The courage to be constructivist. *Educational Leadership, 57*(3), 18–24.

Clarke, N., Davies, P., & Skeers, R. (2005). Self and peer assessment in software engineering projects. *Proceedings of the 7th Australian conference on Computing Education - 42 CRPIT '42*, 91-100.

Clarke, N. (2002). Software engineering projects: Working in teams. *Proceedings of the 6th IASTED International Conference on Software Engineering and Applications, Cambridge, USA,* 698-703.

Clarke, N. (2005) Evaluating student teams developing unique industry projects. In A. Young & D. Tolhurst (Eds.), *Proceedings of the 7th Australian Conference on Computing Education (ACE2005)* (pp. 21-30). Newcastle, Australia. CRPIT, 42. ACS.

Clear, T., Young, F., Goldweber, M., Leidig, P., & Scott, K. (2001). Resources for instructors of capstone courses in computing (Working Group Report ITiCSE). *SIGCSE Bulletin, 33*(4), 93-113.

Crebert, G., Bates, M., Bell, B., Patrick, C. J., & Cragnolini, V. (2004). Developing generic skills at university, during work placement and in employment: graduates' perceptions. *Higher Education Research and Development, 23*(2), 147-165.

Conn, R. (2004). A reusable, academic-strength, metrics-based software engineering process for capstone courses and projects. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Norfolk, Virginia, USA,* 492-496.

Daniels, M., Faulkner, X, & Newman, I. (2002). Open ended group projects, motivating students and preparing them for the 'real world'. *Proceedings of 15th Conference on Software Engineering Education and Training, 2002 (CSEE & T 2002), Covington, Kentucky, USA*, 115-126.

Dart, P., Johnston, L., & Schmidt, C. (1996). Enhancing project-based learning: Variations on mentoring. *Proceedings of 1996 Australian Software Engineering Conference (ASWEC 96), Melbourne, Australia,* 112-117.

Dart, P., Johnston, L., Schmidt, C., & Sonenberg, L. (1997). Developing an accredited software engineering program. *IEEE Software*, *14*(6), 66-70.

Dawson, R. (2000). Twenty dirty tricks to train software engineers. *Proceedings of the 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland*, 209-218.

Falkenberg, L., Russell, R., & Ricker, L. (2000). Linking theory with practice: undergraduate project management with school-age children. *Journal of Management Education, 24*(6), 745-762

Fincher, S., & Petre, M. (1998). Beyond anecdote towards real transfer: Using other institutions' experience of project work. *Proceedings of the Annual Joint Conference Integrating Technology into Computer Science Education: 6th Annual Conference on the Teaching of Computing and the 3rd Annual Conference on Integrating Technology Into Computer Science Education: Changing the Delivery of Computer Science Education. Dublin, Ireland*, 86 – 89.

Fincher, S., Petre, M., & Clark, M. (2001). *Computer science project work: Principles and pragmatics.* London: Springer-Verlag.

Fleming, S.T. (2005). Talking past each other – Student and staff reflection in undergraduate software projects. *Issues in Informing Science and Information Technology*, *2*, 92-102. Retrieved from http://proceedings.informingscience.org/InSITE2005/I08f58Flem.pdf

Garlan, D., Gluch, D. P., & Tomayko, J. E. (1997). Agents of change: Educating software leaders. *IEEE Computer, 30*(11), 59-65.

Gehrke, M., Giese, H., Nickel, U. A, Niere, J., Tichy, M., Wadsack, J. P., & Zundorf, A. (2002). Reporting about industrial strength software engineering courses for undergraduates. *Proceedings of the 24th International Conference on Software Engineering. 2002, Orlando, Florida,* 395 – 405.

Goold, A. (2003). Providing process for projects in capstone courses. *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education, Thessaloniki, Greece*, 26-29.

Greening, T., Kay, J., Kingston, J. H., & Crawford, K. (1997). Results of a PBL trial in first-year computer science. In J. Hurst (Ed.), *ACM Proceedings of the 2nd Australasian Conference on Computer Science Education,* 201-206.

Hribar, M. (2005). Sure fire programming: A general framework for independent projects in computer science. *Journal of Computing Sciences in Colleges, 21*(1), 257 – 266.

iParadigms. (2007). TurnitIn homepage. Retrieved 23 October 2007, from http://www.turnitin.com/static/home.html

Kolb, D. (1984). *Experiential learning - Experience as the source of learning and development.* New York: Prentice Hall.

Kolb, A., & Kolb, D. (2005). Learning styles and learning spaces: Enhancing experiential learning in higher education. *Academy of Management Learning and Education,* 4(2), 193–212.

Krause, K. (2006). Dimensions of student engagement: New opportunities for learning and teaching [Keynote address]. *The University of Ballarat Learning and Teaching Conference*, *Ballarat, Australia.* July 2006.

Linder, S. P, Abbott, D., & Fromberger, M. (2006). An instructional scaffolding approach to teaching software design. *Journal of Computing in Small Colleges,* 21(6), 238-250

Lynch, K., Goold, A., & Blain, J. (2004). Students' pedagogical preferences in the delivery of IT capstone courses. *Issues in Informing Science and Information Technology, 1*, 431-442. Retrieved from http://proceedings.informingscience.org/InSITE2004/067lynch.pdf

Lynch, K., Heinze, A., & Scott, E. (2007). Information technology team projects in higher education: An international viewpoint. *Journal of Information Technology Education, 6,* 181-198. Retrieved from http://jite.org/documents/Vol6/JITEv6p181-198Lynch354.pdf

Mann, S., & Smith, L. (2005). A+, Guaranteed: Insisting on best of practice within capstone projects. *Proceedings of the 18th Annual Conference of the National Advisory Committee on Computing Qualification*, *Tauranga, New Zealand,* 243-248.

Martin, E. (1997) *The effectiveness of different models of work-based university education.* Retrieved April 21, 2006, from http://www.dest.gov.au/archive/highered/eippubs/eip9619/front.htm

McLay**,** A., Corich, S., & Millma, M. (2005). Projecting students to employment. *Proceedings of the 18th Annual Conference of the National Advisory Committee on Computing Qualification, Tauranga, New Zealand.* 265-268.

Newman, I., Daniels, M., & Faulkner, X. (2003). Open ended group projects: A 'tool' for more effective Teaching. *Proceedings of the Fifth Australasian Conference on Computing Education Adelaide, Australia*, 95-103.

Nicol. D. J., & Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: A model and seven principles of good feedback practise. *Studies in Higher Education, 31*(2), 199-218

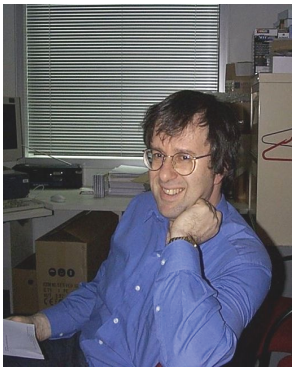
Nunan, T. (1999). Graduate qualities, employment and mass higher education. *Cornerstones: Higher Education Research and Development Society of Australasia (HERDSA) 12-15 July, 1999. Melbourne, Australia*.

Purvis, M., Purvis, M., & Cranefield, S. (2004). Educational experiences from a global software engineering (GSE) project. *Proceedings of 6th Australasian Computing Education Conference (ACE2004), Dunedin, New Zealand,* 269-275.

Tam, M. (2000). Constructivism, instructional design, and technology: Implications for transforming distance learning. *Educational Technology and Society, 3*(2), 50-60.

Biographies

Kathleen Keogh is a lecturer in computing at The University of Ballarat, Ballarat, Australia. Kathleen has taught capstone project units over the past ten years, both at the University of Melbourne and currently the University of Ballarat. Her research interests include teaching of student projects, teamwork and the coordination capabilities of artificial agent teams in simulations of complex situations.



Leon Sterling is Director of E-Research at the University of Melbourne, and also Professor of Software Innovation and Engineering in the Department of Computer Science and Software Engineering. He previously worked in universities in the UK, Israel and the USA. His teaching and research specialties are software engineering, agent technology, logic programming, especially the language Prolog, and artificial intelligence.

Anne Venables lectures in Computer Science at Victoria University, Melbourne, Australia. She has research and teaching interests in artificial intelligence and intelligence systems. As a former secondary Science and Mathematics teacher who has migrated into tertiary education, Anne is also interested in innovations in education and has previously published in this field.