Projecting Projects: Choosing Software Engineering Projects

Lesley Smith **Samuel Mann**

Department of Information Technology and Electrotechnology
Otago Polytechnic, Dunedin, NZ
lsmith@tekotago.ac.nz

There is a wealth of literature about the incorporation of clients with real business problems in software engineering courses. There is, however, little by the way of direction in the selection of projects. In this paper we analyse six years of experiences of teaching software engineeering using a client based approach. We develop guidelines to aid in the identification of clients and projects. This process needs to deliver both software engineering theory and provide a platform for later capstone projects. The paper describes each iteration of the course including content analysis of student reflective reviews. The paper concludes with ten point guidelines.

1. INTRODUCTION

In a vocational IT degree a second year course in software engineering has a dual role. It needs to encompass software engineering concepts and prepare students for the capstone project. These two roles are not necessary complementary (Goold 2003). A common practice is the use of a real business case to anchor such a course. Students undertake a development for these clients, following a prescribed development process, accompanied by theoretical instruction. The intention is that the students experience the scope of software engineering with all the implicit difficulties: client issues; complexity of business systems and group work. Chamillard and Braun (2002) argued that "the most critical aspect of the (software engineering/capstone) sequence is the use of real projects, with real customers" (p227).

Basing the course strongly around real clients with a real business problem poses extra difficulties for the instructor, not the least of which is finding and managing the client. This is, we argue, perhaps the most crucial decision in such a course, the choice of example project.

2. BACKGROUND

The incorporation of a real world project into a software engineering course is not new. Collofello and Woodfield (1982) described the teaching of a four semester software engineering course, using a project for "unification:

> "The project would be of medium-size (3-5 man years) and as such would be a more faithful reproduction of the real world. The experience gained from participation in a project is very beneficial. In addition to being realistic (as compared to toy programs found in most computer science classes) such projects also promote better team work" (p14).

Collofello and Woodfield described the benefits of an external client as "more realistic" but saw "drawbacks because the instructor does not have total control over the project" and the logistics were too onerous. They used in-house projects, at least "until we have gained more experience" (p17). By 1989 Northrop was able to describe several courses using real projects as both meeting academic objectives and providing "more than an academic exercise".

Gabbert and Treu (2001, p191) argued that "few would argue with the assertion that it is beneficial for students to work on solutions to meaningful problems – solutions which will actually be used by real clients – as opposed to rudimentary or 'toy' problems". They go on to ask "how can such projects be effectively introduced?" (p192). They argue that issues such as the size and scope of real projects, time, finding clients, repeatability, and finding real projects that meet pedagogical objectives of the course, "seem to present insurmountable obstacles to the use of real world projects" (p192). They then go on to recount an example of successfully incorporating a real client, a web-application design, and conclude "we have found that given appropriate subjects and appropriate strategies for teaching those subjects, real projects can be used very effectively" (p197). Unfortunately they do not attempt to generalise their experiences by identifying such appropriate subjects or strategies.

Beasley (2003) describes the "nuances" of making a project run smoothly, one of these is the establishment of a "project pool" (p124) and to "weed out the projects that are inappropriate for such an experience" (p125) but other than eliminating simple web pages, his advice is based on a system interacting with "at least two database tables – preferably three or four" (p125).

Stein (2002) described what he considered to be important factors in quality projects. He discussed the impact the type of client had on learning, for Stein successful projects came followed when "the customer had a clear notion of what they wanted, stated as clearly specified requirements" (p4) whereas "a customer who did not really know what they were looking for – the requirements changed out from under the team regularly. Such projects, although they usually got students passing grades, were not successful" (p4).

Chamillard and Braun (2002) discussed the effect of the IT knowledge of the customer: too knowledgeable led to requirements of the type 'this is how you should do it' rather than 'this is what I want done'. Customers without IT knowledge on the other hand were unable to help with an important formula. While this conflict meets the criteria of being 'real', Chamillard and Braun recognised their best project as one where the customer was "knowledgeable without directing the students" (p230).

Clear *et al*. (2001) suggested that "given a lucid set of course goals it should then be possible to characterise appropriate and (possibly even more important) inappropriate capstone projects". In answering the question "what are potential characteristics of a capstone project?" they list many factors starting with "software development is required" but then the factors become about process: "students must work in teams", "team-sponsor interaction must be professional and realistic" and so on. Clear's consideration of who conceives projects leads to a discussion of the various benefits of student initiated projects, instructor developed and external projects among others. They then argued that despite a higher success rate for capstone projects, "it is necessary to make sure that the project is not critical to the sponsor" (p96), this they argue, is too high a risk for the sponsor and the "undue focus on production of deliverables…will almost always be in conflict with the broader educational goals".

There are obviously many papers describing the teaching of software engineering. Most of these papers are anecdotal, describing what the author did and how it worked, indeed this paper is no exception. Here we focus on the characteristics of the client and the resultant project, aiming to identify guidelines for this crucial component of the course.

3. METHOD

The authors teach Software Engineering, a compulsory paper taken by second year students in a vocational IT degree. This course develops an understanding of software engineering entailing knowledge of the methods and problems of the development, implementation, and management of information systems. The focus is on data-centred analysis, modelling and design techniques as embodied in the Systems Development Life Cycle (SDLC, Hoffer *et al.* 1999) with an added focus on prototyping. Students do not implement the systems they develop.

The whole class (usually two streams) works with the same client but independently in groups. Students self select into groups to undertake the project which takes the whole of a semester long course (See Smith this volume).

We review seven iterations of the course, each with a different client. Material is taken from moderation records, student reflective reviews and course materials. Student quotes are used verbatim but with identifying material removed. It is hoped that this framework will provide a structure for examining the different approaches while allowing flexibility for emergent themes.

4. PROJECTS

4.1 Ship safety management system

The ship safety management system was developed with a client who is a master of an offshore supply vessel.

It was chosen in the belief that none of the students would know anything about the application and would, therefore, be entirely reliant on the development process.

> "Dataflow diagrams proved to be a major learning hurdle for us…this was possibly related to our lack of knowledge of the safety system, …this was overcome by persistence, lots of checking with the lecturer and eventually accepting that the DFDs were not perfect, but had served their purpose of making us think about the processes in the system" (LS2)

A development methodology provides a pathway for developers take a project from vague concepts to detailed code. It was hoped that the selection of this project, with a potentially huge scope would also emphasize the role of a formal methodology in providing a pathway:

> "When we first looked at the brief for Captain Black we thought the that scope for the project had the potential to be much larger than anything we could confidently develop" (LS2)

The project was ideal for covering the scope of the SDLC. The client had a business problem, that of safety at sea and didn't really mind where the project led. Some groups interpreted the goal as a training system, others as an emergency procedure log, and others as a live monitoring system. This variation gave many possibilities for discussion. The maritime environment also led to interesting twists for task analysis and logical design.

This iteration of the course was reported in Mann (2000) as being empowering and considered successful:

> "Despite being a long way outside their comfort zone, groups used the tools of software engineering to design industry strength and innovative systems. Their work was stunning and they can be proud of it".

4.2 Job management system for small engineering business

Three separate clients were identified with very similar needs: a desire for a job management system for small engineering businesses. This was an interesting and very real project that is likely to be continued as a capstone project. The local clients were accessible and got lots of ideas for improving their businesses (*pers com* 2004).

The intention in exposing the class to three clients was that the students would be able to develop generic software that could be delivered to not just these clients but also others in similar businesses. This was unsuccessful. The three different clients worked in very different ways meaning some student groups did not get enough exposure to the needs of the clients: "understanding of what was happening within the business as well as his own commitment to its process was patchy" (RS226), "getting blood out of a stone (but) he cannot be blamed for not knowing what kind of system he wanted" (DR2), "clients giving vague descriptions of what they want are probably very common in the real world, this is why I enjoyed this project" (MG3). Other groups suffered from excessive client involvement, the client came in to see the students most weeks, and as his enthusiasm grew he appeared to be changing his mind on what he wanted almost daily.

The different experiences also made it hard to teach the theoretical aspects of the course with groups at very different stages.

With the exception of project identification and selection (project was pre-selected), the projects lent themselves to a full scope of the SDLC. It did not prove to be exciting. The project was ideal for data process steps (DFD and ERD), however, a problem for this project was the significant proportion of the class who thought they knew about the issues of running a small business. For them the lecturers' insistence on following a convoluted development process was a hindrance: "we were guilty of suggesting a solution at the start and then basing a project around trying to meet that goal" (GE1).

For others, the problems of the client business threatened to become overwhelming "these are worthy goals but not achievable within the constraints and certainly not a discipline that could be introduced into XYZ Engineers …we agreed that the purpose of IT205 was for us to learn the process, not fix XYZ Engineers" (RS226).

The difficulty of clients led to many interesting student comments regarding the structure of the course: "why are there so many different outcomes to the same problem…I felt the lecturers let down the process by not doing enough homework on what was going to result…shouldn't the lecturers have get a tighter reign" (DR2). Lecturer comments at the time "it was interesting to see the groups using the SDLC to create some order from chaos…we need to make it clear that students are marked on ability to demonstrate the process, not on actual outcome". Some saw the difficulties of clients as a challenge: "I'm also kind of happy we got a client like this, I don't know why; I think it gives us more of a feel of the things that could go wrong next year" (RL2).

4.3 Student management system

The student management system (SMS) was chosen at a time when the institution was investigating the development of a new system. It was doing this in a high profile manner and as such the students were aware of many issues. The client was the manager of the institutions Management Information System (MIS) who was very keen to have the students work with him. This worked well on two fronts, not only was he available, he was knowledgeable about the IT development process. He managed, however, not to impose his computing knowledge. This was a successful balance.

This project was interesting as the students initially thought it too small – surely it is a glorification of the three entity student:enrolment:subject relationship widely used in teaching. An SMS is, of course, a huge undertaking. This realisation was drawn out by the development process. It became a challenging project.

The SMS is a traditional data driven information system, ideally suited to the SDLC, especially the data modelling sections. The limitation was the size of the project, there were so many business rules to be captured that the students couldn't possibly do a good job.

This iteration of the course was described in Smith *et al.* (2001). In this year we made the students swap projects after the analysis ("Run over by a bus"). Although we recommended this practice, we have not been so blatant in manipulation since.

4.4 Learning and motivation software

The decision to work with a motivational expert was prompted by a desire to introduce a project that was less defined and had more potential for creative work than previous data driven systems. Some very exciting systems were developed and students successfully learnt software engineering: "the time spent designing and building this system was well worth it just to learn the stages of software development".

This was a perfect exercise for teaching the importance of early stages of development. The client had little idea of what he wanted, only recognised that he would benefit from exploring such developments to "be a global player and have more free time". It also was good in highlighting to students the value of SDLC stages in progressing the project:.

> "There is no one word that can summarise my experience with IT205, the material taught was justified and necessary, the time, subject matter (the motivation system), and expectations were challenging to a point where it seemed almost unreasonable. In saying that I believe it is either intentionally or unintentionally made that way to give the students experience with the kinds of pressures and constraints applied to IT specialists in the real world. In that respect it was 100% successful" (BM1)

Some students however continued to struggle with the project and this impacted on their understanding of software engineering:

> "Client information that was, to say the least, abstract. As a group we were constantly unsure of what we were trying to achieve and how to progress…we floundered when it came to detail and I failed to understand the SDLC given a different client" (PI2)

This project excited students but as it progressed they found it frustrating, as it was so ill defined.

Poor groups scoped this project very small, to the extent of developing little more than login systems. The lack of scope did mean that more able students were able to push this project in directions that proved exciting and innovative. This project led to employment for two of those students to implement portions of the system. The remainder of the system has been a suggested capstone project for two years, but, to date, no project groups have undertaken this.

4.5 Remote advisory system for engineering

In this iteration we attempted to identify a business opportunity for development, rather than solving a specific business problem for a particular client. The nominal client was a business consultant (who taught some of our business papers). This approach did not work well. The identified opportunity was the need of isolated garages to have access to specialised knowledge. This was potentially a nice mix of physical and software development. The class wrote themselves an initial brief.

The nominal client concept did not work. Students identified early on that it was not real and their development work reflected this: “I never really thought that the project could ever really happen in real life” (AH1). Many solutions were not sensible – a fleet of helicopters was needed for one and another did away with mechanics altogether and replaced them with remotely operated robots. Without being real, the students were more tempted to fake the analysis: “alternatives…I feel our biggest mistake was in not doing this step correctly, as we pretty much fudged the results to fit in with what we wanted to do”. Also, strangely, the client discouraged groups from developing prototypes which meant the bizarre ideas were never really tested.

Students found this project too hard. Many students felt that they could not do the project as they “could not understand the workings of how the mechanics of the engine worked”. For some reason this problem became paralysing for several groups despite the actual mechanical information being at most tangential to the project.

During analysis we arranged for groups to visit local garages, not for mechanical information but for understanding of the mechanic work processes. We also started to make use of the several students in the course who had previously been mechanics: “to have someone who could understand and ask the correct questions and understand the answers from our interviews with a mechanic” (MY1).

The project worked well for the scope of the SDLC. It was particularly useful for exploring logical design, both tasks and environment “interaction cases were fun to come up with and very helpful in different scenarios…problems such as theft and detection had not occurred to us until they were done”, and design theme “this took a little thinking about, but once I convinced the others that the system was a tool, not a toy, it became much easier”.

One group continued with this project for their third year project, developing a content management system for detailed mechanical instructions.

4.6 Risk management and ethics system for crown research institute

This project was chosen in response to the open ended Motivation project – we wanted a project that was tightly defined. The student groups worked with a client from a crown research institute to develop an animal tracking and ethical approval system. This was to integrate with existing systems. This development suited the scope of the SDLC and was ideal for the data analysis phases as there was much extant processes and information:

> “the generation of alternatives was an enjoyable task because it allowed for our creative sides…or ‘animal-track-n-trace’ system which would seamlessly fit in all the institute’s users” (CA4)

At this point the client pulled back all these ideas and as the project progressed it became apparent that this project was quite limited – it was really tweaking an existing system.

The client for this project was physically remote and initial meetings were via teleconference. The client was a software engineer and gave very thorough feedback on initial stages but this was largely on form rather than on the content. As an IT person he offered to view all the analysis documentation – we normally filter it for non-IT people – and was swamped by the amount of documentation and was difficult to contact after that. Fortunately the groups had already arranged to visit a research station and were able to proceed by creating fictitious users based on the scientists they had observed, although this did cause frustrations: “unsatisfied at some stages…main frustration, was limited contact with the client…direct observation would have been useful…weren’t really involving the end user” (LT2).

Having a client knowledgeable in IT who commented on details of form also led to a focus on the artefacts rather than the value of using the process as a means to understanding. In the personal review one student remarked “I think we should have spent more time tidying the diagrams and less time arguing about the system” (EK3), perhaps missing the point of the development process.

4.7 Maritime museum

The client from the maritime museum came to us with a request for a webpage. The lecturers could immediately see that there was much more to his needs than that. Like the SMS project, this task appealed as it turned out to be much bigger than students' initial understanding. The complexity of museum data, the integration of many existing systems, and the potential for multiple directions meant that very quickly the students realised that without a firm development methodology they would be swimming.

The client was very keen to be involved and was happy for different groups to go in quite different directions. He was able to be involved in a creative manner for the different groups whether they were developing a management system, public genealogy systems or exhibition interfaces intended for children. The very different approaches of the groups gave much material for teaching.

"So from the start, we knew we were looking at a database, but I don't think we quite incorporated the web idea until we knew he wanted free advertising, and that he wanted more people to know what they had to offer. To us, the web idea was an obvious solution. After completing the functional requirements, our concept started to evolve, as we knew he didn't want an online database because he thought people might damage his information. We took this into account… …we thought the way to reach out to the larger/global community was to offer them something they didn't have to travel for, but still enable the museum to profit from" (DS4).

A system that integrates many of the ideas of the different groups was implemented by one of the students as part of a summer scholarship.

5. CONCLUSION

From these experiences we propose the following guidelines.

A project should:

1. facilitate teaching the structure of the chosen methodology (eg SDLC stages, milestones). For early stage developments the client should have an idea of a business problem, but not a solution.

2. facilitate teaching a range of tools and techniques. The more creative projects are better for logical design work but are difficult to apply to data modelling.

3. be real

4. be exciting and interesting (a bonus)

5. be of value to the client (even though is not actually being implemented)

6. have a client who is interested, knowledgeable and available, but all of these in moderation. It doesn't matter whether the client is an IT person or not.

7. provide a challenge for high achievers and be achievable for others

8. start out *seeming* either very large (and use the process to constrain) or very small (and use the processes to explore hidden complexity)

9. actually be large enough to need scoping down (giving the students the experience of managing scope) but not so huge that it is obvious that we don't expect a successful outcome.

10. provide an opportunity for creativity while keeping groups in the same ballpark for teaching the theoretical aspects. Different projects from the same starting point is a good outcome.

11. agile

An area where all our projects were limited is that of maintenance. The only project that was enhancing existing computer systems (Animal ethics) was not successful. The project continuity work of Walker and Slotterbeck (2002) seems promising here.

Core concepts of software development: iteration and incremental commitment are difficult to achieve in an academic setting. When we enforced iteration, students thought that they were being punished; failure to progress through stages is seen to be academically crippling.

Some clients had a business problem for which they thought the solution was a webpage. Despite Clear *et al.* (2001) calling this "technocratic arrogance", our best projects were where we could take the clients' initial ideas and see a bigger side for development. The trick of course is to still deliver what the client originally wanted. We agree, however, with Clear who appeared to advise against instructor acting as sponsor and mentor: "can be torn between the differing obligations of these two roles" (p99). We go further and suggest that using an academic in the client role does not give students the required reality buzz.

The trick of using real clients in teaching software engineering is excitement without dominating software engineering. We think we have got it about right:

> "Now I am no longer worried about how I will do my project but rather what my project will be about" (MG3).

In undertaking this review we came across a wealth of information, particularly in student self reviews. We have fascinating material about group processes, task splitting and crisis management. We recommend reflecting on the material for other courses.

Acknowledgements

Six years of clients and students, also the help of other IT205 lecturers Russell Hynd and Linus Turner. This review was carried out under Otago Polytechnic Category B ethics approval.

References

Beasley, R. E. (2003). "Conducting a successful senior capstone course in computing." The Journal of Computing in Small Colleges 19(1): 122-131.

Chamillard, A. T. and K. A. Braun (2002). The Software Engineering Capstone: Structure and Tradeoffs. Proceedings of the 33rd SIGCSE technical symposium on computer science education, Cincinati, Kentucky.227-231

Clear, T., F. H. Young, M. Goldweber, P. M. Leidig and K. Scott (2001). "Resources for instructors of capstone courses in computing." ACM SIGCSE Bulletin 33(4): 93-113.

Collofello, J. S. and S. N. Woodfield (1982). A Project-Unified Software Engineering Course Sequence. SIGSCE, Indianapolis, Indiana.13-19

Gabbert, P. and K. Treu (2001). "Reality Check: working with meaningful projects in and out of the classroom." The Journal of Computing in Small Colleges 17(2): 191-198.

Goold, A. (2003). Providing process for projects in capstone courses. Proceedings of the 8th annual conference on Innovation and Technology in Computer Science Education, Thessalonki, Greece, SIGCSE ACM.26-29

Hoffer, J. A., J. F. George and J. S. Valacich (1998). *Modern Systems Analysis and Design*. Reading USA, Benjamin Cummings.

Mann, S. and N. Buissink-Smith (2000). What the Students Learn: Learning through Empowerment. 13th Annual Conference of the National Advisory Committee on Computing Qualifications, Te Papa, Wellington, New Zealand, NACCQ.213-220

Northrop, L. M. (1989). Success with the project-intensive model for an undergraduate software engineering course. SIGCSE, Louisville, ACM.151-155

Polack-Wahl, J. A. (1999). Incorporating the Client's Role in a Software Engineering Course. SIGCSE'99, New Orleans, ACM.73-77

SMITH, L., MANN, S. and BUISSINK-SMITH, N (2001) Crashing a bus full of empowered software engineering students New Zealand Journal of Applied Computing and Information Technology 5(2):69-74

Stein, M. V. (2002). "Using large vs. small group projects in capstone and software engineering courses." The Journal of Computing in Small Colleges 17(4): 1-6.

Walker, E. L. and O. A. Slotterbeck (2002). "Incorporating realistic teamwork into a small college software engineering curriculum." The Journal of Computing in Small Colleges 17(6): 115-123.