Concise paper

Technical complexity of projects in software engineering

Dr Samuel Mann **Lesley Smith**
Department of Information Technology and Electrotechnology
Otago Polytechnic, Dunedin, NZ
smann@tekotago.ac.nz

ABSTRACT

This paper explores the effect of project size and complexity on student learning in a project based software engineering course. Different streams of the same course were allocated similar projects but with differing complexity. The smaller treatment did lead to "smaller" projects but this could be attributed to difficulties in stating the business opportunity.

1. INTRODUCTION

Various authors have presented approaches for incorporating real projects into a software engineering course (Chamillard and Braun 2002, Gabbert and Treu 2001). Guidelines for choosing the example project have to date left unaddressed the issue of the optimal size and technical complexity of the project.

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).

Mann and Smith (2004) examined seven iterations of a software engineering course and extracted guidelines for selecting projects for use in teaching:

A project should:

1. facilitate teaching the structure of the chosen methodology (SDLC, especially each stage, milestones). For early stage developments the client should have an idea of a business problem, but not a solution.
2. facilitate teaching each of the methodologies' (SDLC) 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
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
9. 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.

The 8th guideline, that relating to the scale of the project is the focus of this paper. We can conjecture that projects of differing size and complexity will have differing effects on student learning. Mann and Smith (2004) commented that "our best projects where were we could take the clients' initial ideas and see a bigger side for development".

A large and complex project can lead to good outcomes, emphasizing the role of a formal methodology in providing a pathway (student quote from Mann and Smith 2004):

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

This, though, can have a disempowering effect, a potentially huge motivation project was clearly not feasible and students lost interest poor groups scoping this project very small, to the extent of developing little more than login systems. This danger was avoided with a large Student Management System that started small but students had the advantage of it being familiar territory.

At the other end of the scale a very small project can further shrink as students lose interest as a result of feeling that the SDLC process is overkill. In the middle of this are tasks that turn out to be much bigger than students' initial understanding. These, Mann and Smith (2004) also favour, "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".

So, we have projects that are too small, just right and too large and this varies during the development and differs according to the whether viewed from the lecturers', students', or clients' perspectives. This is not much help as far as guidelines go. The aim of this paper is to explore the effects of the size and complexity of software engineering project on student learning.

3. METHOD

Separate streams of a software engineering course were presented with different projects of differing technical scope. Both projects however, have a booking system at the core, meaning that the same development process could be followed. The 'small' project was a video shop - initially a simple hire system ("VideoShop"). The larger project was a management system for a leisure complex requiring space and time based bookings, staff allocation and invoicing ("Pool"). Both projects had eager clients happy and available to talk with students and appeared equally IT literate.

The different learning paths for the groups within the treatments are examined using an emergent themes approach, student quotes are used verbatim but with identifying material removed.

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.* 2002) with an added focus on prototyping. Students do not implement the systems they develop. Each stream worked with their client but independently in groups. Students self select into groups to undertake the project which takes the whole of a semester long course. The two streams came together weekly for lecture material. Five teams undertook the Pool project and three the VideoShop.

4. RESULTS

The patterns of learning for the different treatments were quite divergent as the classes progressed through the SDLC process.

The initial identification of the projects was carried out by the lecturers rather than the student groups; however the students were required to identify the business problem/opportunity. The effect of the complexity of the task was compounded here by the ability of the client to describe the problem. Despite being considered the larger treatment the problem statement for the Pool, could be clearly stated by students:

"The present booking system for Moana Pool is by manual entry in to a diary and relies heavily on the knowledge of the present Assistant Manager…the recent redevelopment has brought about a large increase in the number of users… there have been problems with double bookings and where there has been a need for cancellations, this has resulted in multiple changes to the diary…the planned booking system will benefit the business by

optimizing water and time management within the complex".

This had the desired effect of seeming so large that the students were grateful for a process such as the SDLC:

"At the onset I had no clue of being able to do what was required, so I didn't have a preconception about what it would look like. I did not think we could do it and definitely not me. Now I see it can be done..."

The VideoShop was similarly a replacement of an existing system, but the business opportunity for this eluded the students. This resulted in meaningless opportunity statements:

"Due to additional requirements needed by VideoShop regarding their current video rental system, an opportunity has arisen to develop a new enhanced system which will meet these requirements. The current system lacks features required by management and staff to progress financially in the business".

Another group stated

"The goal is to attract more customers to make more money...(that) can only be achieved by developing a successful system for the store...tracking movies is inefficient...with redundancy in tasks...more pleasant as interactions more efficient".

In the second stage of the SDLC, students initiate and plan the project. This entails establishing a management document for the team. This includes statements about the work estimation but these are not reliable. The groups knew they had a semester to do the work and carefully managed their work-plan to fill the time available. Perhaps more revealing at this stage are differences in risk management plans. The Pool groups clearly identified economic, technical and ethical feasibility and identified the risks associated, whereas the VideoShop groups, still unsure as to what their project was, worried:

"the company we are designing for is not sure what they want and do not necessarily believe they need an upgrade to a new system".

In the analysis stage the groups undertook requirements research (interviews, role plays, research), requirements structuring and the generation and validation of functional requirements. The Pool groups described the work process and could easily see areas of inefficiencies and possible improvements. Existing documentation was examined and most of the groups took the opportunity to examine alternatives and visit local sites running other resource booking systems. There was in fact an existing information system that performed many, but crucially not all, of the clients needs.

During this stage these groups were reminded by the client that while simple to describe, the Pool management system was a significant undertaking. Her interactions with the groups were littered with statements such as "so many combinations for the pool, it's quite hard", or "so you can see why this takes all morning", and "I need to be able to tell the system: 'yes I can do it, now you (the system) help me".

During analysis two stronger groups began to explore the complexities of time/space relationships in booking systems. During interviews with the client they canvassed potentially difficult areas in the booking structures such as "we need the 6 lanes of the shallow end every Monday morning at ten, except Labor day when we'll come on Tuesday and on the last day of term we need the wave pool as well". Exceptions were explored, such as "we don't put canoe polo in beside the aqua joggers" and the potential for capturing expert knowledge "I always know to be careful in term 4, because that's when School X comes but they never get back to me until the last minute". At this stage the complexities of the booking process were explored with ideas of the need to represent tentative pencil bookings and allowing clashes but flagging them as needing to be resolved.

The clear work processes leant themselves well to being described by process diagrams and entity relation diagrams. Most ERDs were quite similar and simple three entity user-group: booking:facility structures.

This stage resulted in most groups developing strong functional requirements. Despite not having to implement the system, most groups limited their functional requirements to something that the group would be comfortable to build.

The analysis stage was a difficult one for the VideoShop groups. Like the Pool, the VideoShop was ideal for interviewing different people and considering various user groups and stakeholders. Also similar, the VideoShop had an existing information system but in this case this was problematic for the students: "it does what it has to but you've just got to know the tricks of the trade like Ctrl-Y to delete".

The poor understanding of the business problem translated into poor functional requirements. This though, provided a platform for progression and increased opportunities for learning. Initial functional requirements had meaningless "allow data entry" and "allow password protection" but progressed through "manage customer information" to "support hiring and return of stock (videos)". Other groups described "what VideoSeller does plus", without any real understanding of either the "does" or the "plus". Somewhat bizarrely the groups could not see any value in carrying out process flow or data modelling tasks, what they delivered were half hearted efforts of a very simple movie:customer nature (ignoring complexity of time:date, item:movie etc).

We directed the groups into extending their possibilities beyond the actual hiring of videos, to include search systems. This enthused the groups for a while, until they found the vast online movie databases, but rather than finding these inspiring, they became disheartened and questioned what they were doing. Eventually the all groups produced suitable functional requirements but this was seen as a defeat rather than a victory:

> *"Almost all the work we did in analysis was irrelevant to the project after the change in scope".*

All three VideoShop groups became distracted by the technology. Prototypes developed to validate the functional requirements described "database" as a solution, but without any notion of how that might solve any business problems. Another group produced a booking website, an information kiosk and a website but all described the same limited set of functional requirements. One group did have an interesting left field suggestion of ADSL video on demand but they didn't take it seriously.

In compiling prototypes to validate the functional requirements we expect students to develop four systems: status quo plus a bit, two reasonable systems and an over the top system. In all cases (except the video-on-demand) for both the Pool and the VideoShop, we (and the client) selected the over the top system for development.

The design stage is an iterative process where teams first focus on the user interaction, starting with task and user analysis and dialog designs and then combining these with design concepts to produce wireframes, these are tested against the task descriptions before final interface designs are produced. These designs are accompanied by parallel developments in the data model, and are followed by the physical design.

Poorer groups struggled with the idea of discarding their functional validation prototypes and spent this stage perfecting infrastructural screens (login etc) but never coming to terms with the complexity of the interaction of the more important functional areas. Three of the Pool groups made this mistake and struggled, although one later realised the mistake and started the stage again.

Both the Pool and the VideoShop gave the opportunity for consideration of complex interactions. For the Pool the stronger groups used long descriptions of test bookings and used these tasks in testing the interfaces. Both of these groups, however, designed systems that could deal with these exceptional cases, but their systems were not flexible or robust outside these stated test cases. The strong groups focussed on different complex aspects of the system. One group examined the aspects of pencilled bookings (temporarily permitting clashes and giving approaches to resolve conflicts). The other worked hard on understanding the complexity of time and spatial arrangement (Figure 1) but although this was contained by the ERD (using a time template) and represented this on the interface, they did not really consider the interactivity aspects – they had a both a map and a clever calendar but didn't really integrate the two.

The struggle to see the purpose of the VideoShop development continued in the design stage. Groups had poor links between the data and the interfaces. Data models were not well developed. One group had a product table inelegantly linked to six others (video, game, hardware etc). Another had 16 tables but except for unneeded movie details (which could be gotten from elsewhere), the core was a product-hire-customer model, with time/date as a single text field in

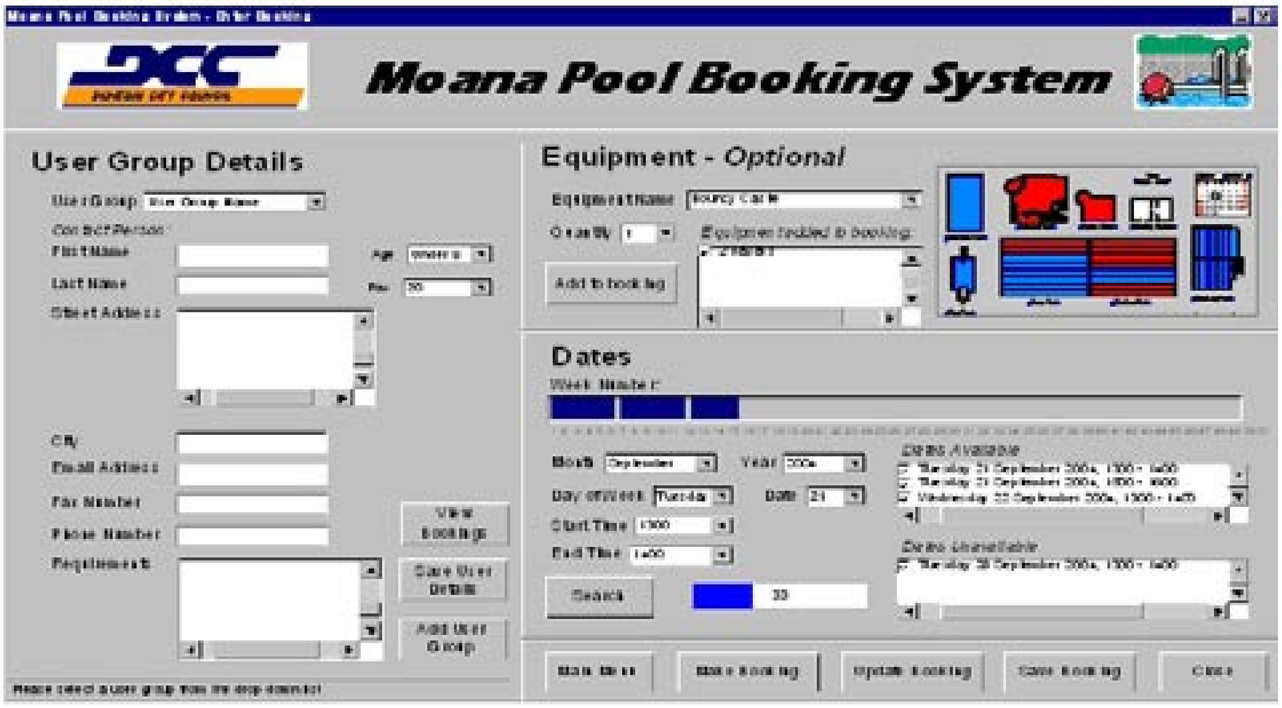

Figure 1: Time and space considered.

Table 1: Treatment results, no differences significant in t-test.

Stream	n	n groups	Project average	exam individual	exam std dev	non project
Video	11	3	77.5	12.76	4.36	35.054
Pool	15	5	66.6	13.25	4.27	33.94

hire. This missed the potential complexity of the item:title relationship.

The theme of being distracted by technology continued for the VideoShop groups. One group did describe complex interactions possible via different configurations (in person, phone booking, web, kiosk), but these were developed entirely independently, with no notion of a modular architecture providing the core structure. Another group proposed an instore video booking system but failed to link this to a stock tracking system. The third developed a kiosk but, without a clear direction as to purpose, did not take into account the interactivity afforded by the kiosk: although they did have "big buttons" they still required a login on the first screen (as if modelled on an ATM rather than a walk-up exhibit). The kiosk designed also ignored their highest priority functional requirement: "manage customer information".

The other potential area of complexity for the VideoShop is the search, but the groups had an overly simplistic view of this. Although one group recognised this as the prime function and made it an always viewable item, the results betrayed their roots in databases. There was little use of related records, push marketing, suggested searches, top ten searches or the like. Despite the client describing in the first interview the needs of customers who want to see the movie "with the guy who dressed up as his children's nanny but this one is about photographs", there was little acknowledgement of having a marketing or customer focussed approach.

The two treatments did not result in any significant differences in marks for either the projects or other assessed components (Table 3).

5. DISCUSSION

This paper has examined the effect of technical complexity on student learning in software engineering. Two treatments were intended to provide similar projects, differing primarily in size and technical complexity. While both primarily booking systems, the larger project (Pool) had potential for more complex interactions, and complex representation of time and space. The VideoShop was prima facie smaller but could have been expanded to include complex interactions. These differences between the treatments were too subtle for the students, even the strongest groups only touched the surface of the areas we had identified as complex. The smaller size project was much harder to state in terms of business problem and it was this that caused "smaller" solutions. These smaller solution, although less satisfying, still got students passing grades (also noted by Stein 2002). In this analysis we have not accounted for differing abilities of students. Further research should include such analysis.

It was not possible to accurately assess the size and complexity of the projects. They are probably about the same size. It would be pointless to attempt to perform function point analysis, entity counts or other such metric as students scale the project to fit the time available. To the students' dismay, the client and lecturers picked the largest of the development options for all but one group.

The small project seemed approachable at first but despite urgings from the lecturers all groups managed to continually shrink the project. The larger project, daunting at first became achievable through early stages of the development and then difficult as underlying complexities became apparent. Most groups realised the complexity but only skimmed the surface.

While providing an interesting exploration, this paper has not managed to provide much further guidance than the Goldilocks approach of picking projects that are not too small, not too big but just right. It has though highlighted the potential importance of the ability to clearly state a business problem/opportunity statement.

Acknowledgements

We thank the IT205 students of Semester 2 2004 who are mostly now half way through their capstone projects. We are also indebited to the industry clients Jason, Steve and Mandy who were all committed, creative and thoughtful. This work considered under Category B Ethical Approval.

REFERENCES

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

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, et al. (2002). Modern Systems Analysis and Design. Reading USA, Benjamin Cummings.

Mann, S. and L. Smith (2004). Projecting Projects: Choosing Software Engineering Projects. Proceedings of the 17th Annual NACCQ, Christchurch, NACCQ. 183-190

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.