Role of the development methodology and prototyping within capstone projects

Samuel Mann **Lesley Smith**

Department of IT and Electrotechnology
Otago Polytechnic
Dunedin, NZ
smann@tekotago.ac.nz

A significant challenge in the design of capstone courses is relationship between process and product. As academics we argue that a strong process will result in a good product but instructors face little direction in the identification of a suitable process. A major issue is that of systematically incorporating prototypes into the software design and development lifecycle. In this paper we have identified a development framework which successfully does that integration. This framework was developed from exemplar projects and provides a best system framework. The framework is also used to explore the development process of weaker projects.

Keywords

Capstone, Methodology, Prototyping, Guidelines

1. INTRODUCTION

This paper aims to explore the role of development methodology in successful capstone projects, in particular the extent, nature and integration of prototyping.

In computing development projects the choice of development methodology is as perennial a subject for debate as is the choice of programming language. A methodology has to be selected for any given project and the selection may be based on a myriad of factors. In computing education, the choice is complicated by the introduction of the teaching imperative. Many educational institutions contain a capstone project whereby students, often in groups, work to complete a significant project. The choice of development process becomes pivotal in a capstone project (Beasley 2003; Chamillard and Goold 2003;. Bridgeman 2003, Clear 2003). The issue we face is finding not only the best approach for development but the best approach for teaching it.

Clear *et al.* (2001) described what they call the "process versus product tension" (p95) in the design of capstone courses. They refer to the decision as to whether the focus of the capstone course should be on the development process, or on the product of that development: a working system. They argue that as the capstone course emulates work as a professional "it is advisable that an acceptable methodology or process be adopted and followed. What specific methodology or adaptation thereof is chosen is less important than the use of some formal process". Chamillard and Braun (2002) describe the problem thus:

> "Given the importance of process in real software development activities, we want to ensure that our students get appropriate exposure to process issues. On the other hand, we also do not want our students to believe that if they follow an appropriate process, it does not matter if they generate a working product! We must therefore also provide the appropriate emphasis on the product the students are developing to ensure it meets the project requirements". (p229)

What then, is an appropriate process for capstone projects? Traditionally, most capstone courses have used a derivation of the Software Development Life Cycle (SDLC, eg that described by Hoffer *et al.* 1999). In recent years, instructional designers have mixed this with components of agile modelling and especially prototyping (Fincher *et al.* 2001).

At Otago Polytechnic capstone students have previously been taught software engineering with heavy focus on software development life cycle (Hoffer *et al.* 1999). In the capstone course they are required to write an essay that considers the most appropriate development approach for their particular project.

Table 1: Classification of projects and grade distribution.

	Grade				
Area	A	B	C	Fail	Total
Total System	3	0	0	0	3
Hardware	4	4	3	4	15
Applied software	1	4	1	1	7
Information system	4	4	3	2	13
Multimedia	3	6	0	3	12
Content management	3	1	2	0	6
Network	1	1	0	1	3
Other	4	1	1	0	6
Total	23	21	10	11	65

These projects almost invariably conclude with a statement to the effect of "we will use SDLC (or some variant) mixed with prototyping".

It is this 'SDLC mixed with prototyping' that we wish to explore. This issue goes beyond computer education: both the scientific and trade literature widely cover issues of prototypes (fidelity, evolutionary/revolutionary etc eg Rudd *et al.* 1996) but a model for the successful integration remains elusive. Hakim and Spitzer (2000) argue that the "challenge is to systematically incorporate prototypes into the software design and development lifecycle".

In this paper we aim to identify a development framework for successful capstone projects.

2. METHOD

This paper employs a mixed method. We first describe the nature of the capstone projects, and then examine some external characteristics looking for indicators of quality. We then explore the development process employed by ten successful projects. From these we identify a development framework. This framework is then used to analyse a full set of projects, this validates the framework and highlights the important factors in applying the framework.

The full set of projects are all the projects completed in the period 2001 – 2003. (n=65) Projects are undertaken by students in groups of two or three, or occasionally on their own. Groups are responsible for finding their client and project although help is often given. The groups self manage the project, including identifying the development methodology. Groups develop their own marking schedule and present to a panel to justify the marks they have given themselves. This panel, consisting of three academics and at least one industry representative, make a final recommendation for the grade. Project documentation includes a self review, and, with confidential information removed, is lodged in the library.

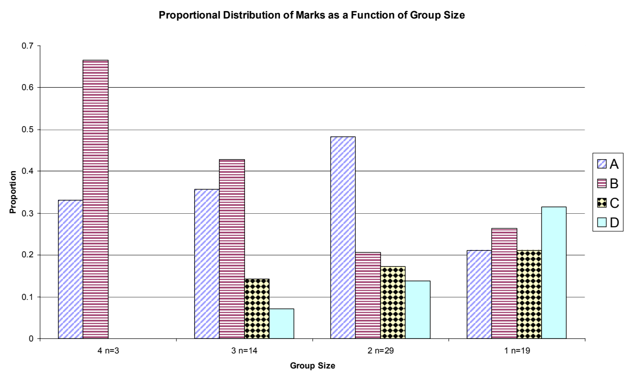

Figure 1: Effect of group size on project grade

3. RESULTS

3.1 Initial findings

It is worth investigating external features of the projects to indicate development characteristics of successful projects.

Since 2001 65 projects have been completed. 35% gained As, 32% Bs, 16% C and 17% failed. The projects can classified into eight categories as shown in Table 1 along with marks distributions for each. With some small exceptions ("total systems" are all successful, 'hardware' has a higher than expected failure rate), there are no strong relationships between the type of project and the grade. Good grades are possible whatever the type of project.

There is an effect of group size, as shown in Figure 1. While groups of two or three show similar patterns, the students working alone have a much higher chance of failure. We believe that there are two underlying factors here: the quality of students (ie why are they working alone), and that they miss the benefits of working in a group.

Most students follow a document centred development approach. The students were wrong however, size does not count, the rumour about assigning grades by weight of documentation is not true. Figure 2 demonstrates a very weak relationship be-

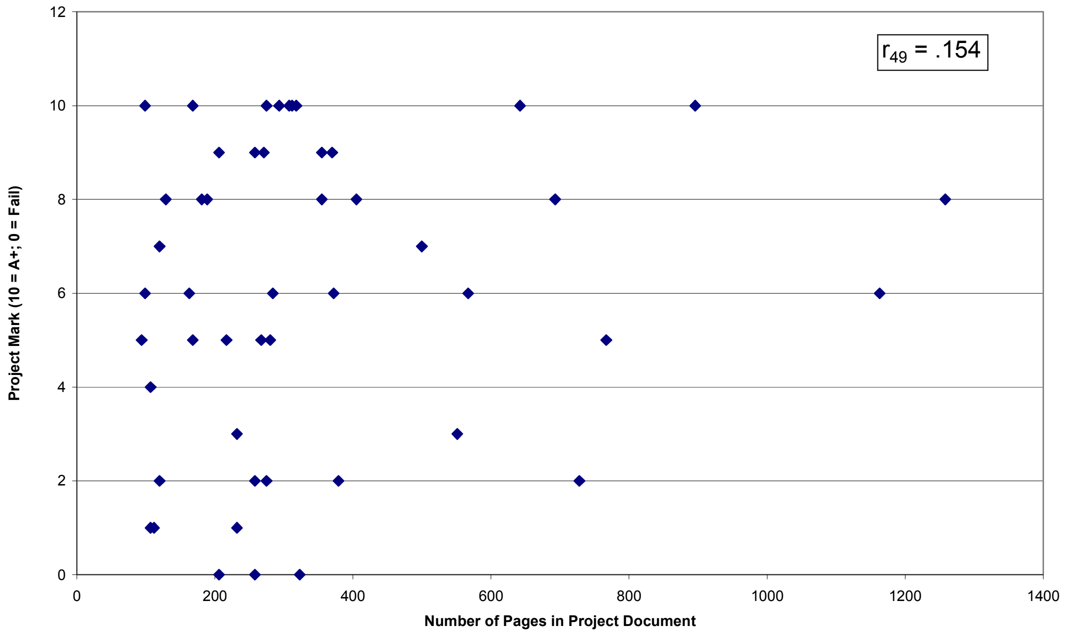

Figure 2: Weak relationship between pages and grade (n=49)

tween the final grade and the number of pages in the documentation (n=49, complete documentation available in the library). Once above a minimum threshold of around 100 pages, any grade is possible. There are a cluster of top grades at around 260 pages. The projects with greatest amount of documentation did not get top grades and clearly poor projects cannot be rescued by a mass of documentation.

Good projects can not be identified by type of project nor by the quantity of work. So we come back to the question, what are the good groups doing?

4. EXEMPLAR PROJECTS

In the following sections we examine the qualitative evidence of the development process of four exemplar projects, the remaining exemplars are briefly described in Table 2. Projects were selected to give a range of project category. Quotes are given from the students' reflective reviews to portray the complexity of the decisions and process.

4.1 Museum

The museum project (Garrett *et al.* 2003) aimed to provide a solution for the Otago Museum. This was a complex project that aimed to resolve the differences between two high level goals of the museum – the long term storage of artefacts that conflicts with the public education imperative. The group developed an automated system that captures 3D objects and displays them on the web.

The group performed extensive prototyping and testing as part of a carefully managed development plan.

A functional prototype involving hardware and software, along with a great deal of 'sellotape' was developed as part of analysis. Despite considerable frailties and faked components this was tested at the museum in order to demonstrate the worth and feasibility of a system as described by the functional requirements. Following a revolutionary pattern the group developed several hardware and software prototypes, each of which they extensively tested at the museum. Doing this enabled the group to perfect the process before automating each step. The group's reflective review describes the excitement of capturing a first museum object (a moa bone), a tiny object (a weta) and a huge object (a stuffed tiger).

Three months before the end of the project the group was able to begin the construction of a production system. This they did to perfection, both the software and hardware systems were extremely well engineered: elegant and commented code and bullet-proof and redundant hardware systems. This system was installed and in production in the museum a month before hand in date.

The group recognised that although they had followed the design stages of SDLC they had perhaps skimped on some sections:

Table 3: Practices identified in exemplar projects used inform development framework

Name	Project Aim	Project type/ Methodology	Application of methodology
ACKI	To create a generic box that sits between computer and keyboard, to feed analogue or digital input from any device, to assist users with limited mobility (Barclay *et al.* 2001).	Hardware Prototype SDLC	• **Stable testbed** was established early • Detailed **testing protocol** established • Detailed log book • "Miniature lifecycles" used for each component
School network	To meet the ICT needs of a local primary school, including network design, implementation and professional development.	Hardware Training SDLC	• Training needs analysis for teachers • Network **prototyped** using a rapid installation • Borrowed equipment for short term to test system
Serve it Right	To assist in staff training at a local restaurant (Ponting *et al.* 2003).	CD Training SDLC Modular approach	• Early **paper based prototypes** used to determine functional requirements • Mocked up the whole system • **Extensive testing** • **Stable platform** led to a strong final product
Post production pathway	To support the extremely complex post production pathway for a television company	Database SDLC prototype	• Complex and detailed DFDs • DFDs were the prototype, **communication tool**, generated FR's and became final product. • **Stable platform** developed • Separate **prototypes** in each stage
Cemetery webpage	To provide a resource for a heritage tourism trust, a dynamic webpage for a cemetery with 18,000 historical records www.canny.co.nz/northerncemetery/	Dynamic database SDLC Prototype	• Developed static webpage as **prototype** for **discussions with client** to develop **functional requirements** but then group "just needed to make prototype robust …"adding functions and polish" • Realised approach wasn't working as client was distracted by detail, and major design decisions overlooked. Restarted logical and physical stages of **SDLC**, changing approach to story based information • Developed **stable platform** early of dynamic webpage functions then built and tested system.
Alternative CAD interface	To develop a drawing board interface to architectural CAD systems (Lawton and Meikle 2003).	Hardware Interface SDLC Prototype	• Iterative stages of analysis, logical design and **prototyping** were run concurrently • "Paper" interfaces used for **user testing** for requirements determination • Cycles of development and configuration of hardware followed by testing • Many different components prototyped
SMS	To control several remote devices via a cell phone	Hardware Software SDLC Prototype	• **Test application** developed to identify design problems • Working system "**prototype**" developed • Prototype used to define **functional requirements** • Replaced major components sequentially in implementation

"Because we spent more time in analysis and began prototyping early we were able to fast track through Logical and Physical Design stages. For these two stages we produced fundamental SDLC deliverables, using them in order to re-enforce the work we were doing during prototyping" however "we did not take into account all the functions required of the software to complete the functional requirements of the system while we were designing the interface, causing us to rebuild the interface at a much later stage. This rebuild of the interface could have been avoided had we done more planning of the interface during the logical design, i.e. wire frames, interface designs".

4.2 Edubuilder

The construction industry is an area where there are multiple layers of complexity, interacting roles, differing timescales and consequences of decisions. This is a prime target for a good interactive learning experience supported by technology. This project undertook to provide a content management system in an area where there is a vast store of expert knowledge available but few good information sources that take advantage of interactive media (Singh-Cosgrove *et al.* 2001) .

The group followed the SDLC and began by working through a huge pile of building standards and struggled to find an underlying theme to hold it together. They used several metaphors before coming back to the building itself and the project plan chart. They developed a diagrammatic prototype and in discussion with several "clients" it became apparent that the different timescales, contexts and user views of a Gannt chart, that this was the elusive concept. To test this premise the group developed several prototypes to prove the concept, these ranged from a single sketch to a semi-functional interactive mockup of the system. These prototypes were used to explicitly test various aspects of the premise.

In order to implement this concept the group realised some complex technical procedures were needed. The group used a carefully managed testing regime that identified a need for a test, the development of code, a test procedure and recording.

As requirements were well defined, a stable system was developed and a rigorous testing regime utilised, the implementation at the end of the project was remarkably rapid yet the deliverable was robust and useful.

4.3 Information system for suit hire

Despite being an ideal candidate for a traditional information system, the client for this project was not entirely convinced that he needed a computerised system. To alleviate this concern the group developed an early prototype on the computer that mirrored the paper based system. The client was convinced that he would gain efficiency while not losing the basis of the paper based system. This prototype also allowed the development of strong functional requirements. The group was also able to test their understanding of the current system by "operating" the prototype in parallel with the existing system, at the clients' premises.

Once analysis was completed, the group discarded the prototype. They then went through normal logical and physical design involving several prototypes. The client was unavailable for five months during this stage but the group was able to test the prototypes according to the strong functional requirements they had developed. The resulting system was substantially different from the original prototype in operation of the business but it retained the look of the paper based system.

4.4 Familtrak

A familiarisation tour ("famil") is a trip taken by people in the travel agency to familiarise themselves with the products they are selling. They are supposed to write up an account on their return but hardly ever do. The client for this project wanted a system to facilitate the reporting and dissemination of such. This was much more than a simple database, the solution involved a webpage, XML, GIS and a high degree of interactivity (Hamilton 2003).

The initial functional requirements were worked out between the group and the client. The group then undertook several weeks of validating and improving the functional requirements. This included much observation of work patterns, a questionnaire and a paper based prototype. A prototype was first used to represent the set of functions the system would be required to carry out. In logical design, a second paper based prototype was used to develop the structure of the system. The group first tested the prototype entirely on paper with the client, and

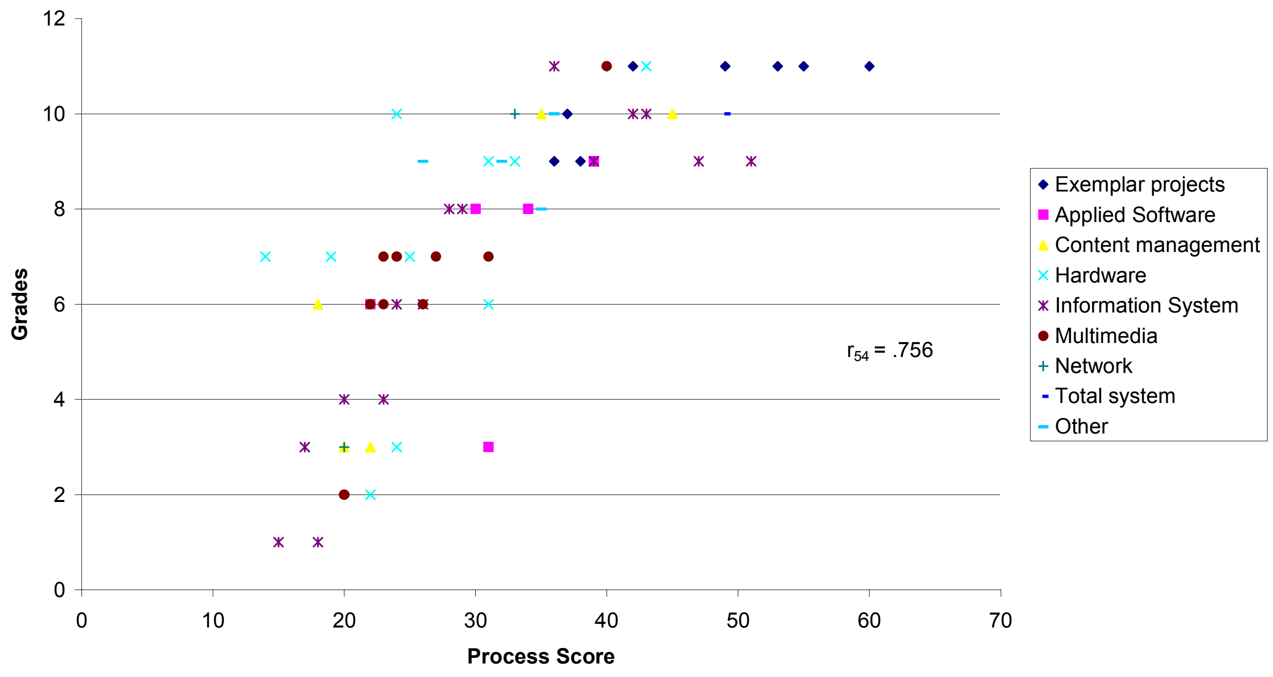

Figure 3: Process score summation is closely related to quality, this is unaffected by the project type.

then in a very clever move scanned in the paper and linked it together, resulting in a low fidelity but high interactivity prototype. This meant that "testing was completed before any code was written".

The product was released to the client for actual production use three months before due date and the group was then able to work on perfecting the system and developing some of the more difficult aspects (linking spatial and multimedia aspects).

5. DEVELOPMENT FRAMEWORK

From the exemplar projects some themes emerge. These we distil into an exemplar development framework.

1. "SDLC" identified methodology although agile in application

2. Prototyping used

3. Strong functional requirements used

4. Functional requirements tested with low fidelity but high interactivity prototypes

5. Stable platform for development developed early

6. Prototypes part of integrated testing plan

7. Early functional deliverable to client

8. Robust final deliverable

9. Still need 'normal' design – prototyping doesn't replace

10. Prototypes used in communication with client

11. Maturation by revolutionary (cf evolutionary). Staged replacement for hardware.

Some of these factors are perhaps expected measures of quality; it is not surprising that they appear here: we did not look at spelling or good intragroup communication but they would be expected on such a framework also. What we have though is evidence that the exemplar groups do these things. Some factors are surprising; we did not expect to find the maturation process so similar for the exemplar projects.

6. VALIDATION

In order to further explore the importance of the development framework, we undertook an analysis of the full set of documented projects. The first author worked through the documentation with a third person who did not know the history of the projects to assign a five point scale to each of eleven factors. These scores are compared to the grades for the projects.

It should be noted that this analysis uses grade used as measure for quality and we recognise that there are other factors at play here. We also recog-

project_id	project_category	project_grade	SDLCBasis	prototype_developed	stongRequirements	testedRequirements	stablePlatformEarly	prototypeUsedsinTesting	earlyFunctionalDeliverable	robustSeconddeliverable	normalDesignStillUsed	prototypeUsedinCommunicaton	Maturation revolutionary	score
95	CM	A+	5	5	5	5	4	5	5	5	5	5	5	53
45	TS	A+	5	5	5	5	5	5	5	5	4	5	5	55
93	CM	A+	5	5	5	4	5	5	5	4	5	5	5	49
79	HW	A+	5	5	4	3	4	4	5	3	3	5	3	40
85	MM	A+	5	4	5	4	4	5	3	5	3	5	3	42
98	TS	A+	5	5	5	5	5	5	5	5	5	5	5	60
1003	OT	A	4	4	4	4	3	5	5	5	3	2	2	37
120	MM	A-	4	3	5	3	4	3	3	5	3	5	4	38
116	IS	A-	5	3	5	5	1	4	1	4	4	5	4	36
58	MM	A+	5	4	4	4	3	4	3	4	5	5	4	40
38	IS	A+	4	4	2	5	4	5	5	2	3	2	3	36
94	HW	A+	4	5	3	2	5	5	5	5	2	5	5	43
92	IS	A	5	5	5	5	3	4	3	5	5	4	4	43
78	HW	A	3	1	4	1	2	1	1	5	2	2	5	24
55	TS	A	5	5	5	5	5	5	5	5	3	5	3	49
49	CM	A	4	4	5	5	3	3	5	5	4	5	5	45
43	IS	A	5	5	5	4	5	4	2	5	5	4	3	42
35	NW	A	4	4	4	5	2	3	3	3	4	1	4	33
1	CM	A	5	4	3	3	4	4	2	4	5	4	2	35
110	OT	A	4	2	4	4	4	2	4	5	4	2	5	36
2	AS	A-	5	5	5	5	2	5	2	4	5	4	2	39
3	OT	A-	4	3	3	3	3	3	3	4	2	3	5	32
46	IS	A-	5	3	4	4	4	2	5	5	3	5	4	39
44	HW	A-	4	4	5	1	3	3	3	3	2	3	4	31
7	IS	A-	5	4	5	5	4	4	5	5	5	5	4	47
174	IS	A-	5	5	5	5	5	5	5	5	5	5	5	51
91	OT	A-	4	3	4	4	1	1	1	4	3	4	1	26
90	HW	A-	4	4	3	2	3	4	2	3	2	5	5	33
60	AS	B+	4	4	5	3	4	3	3	2	3	3	4	34
41	AS	B+	4	4	2	4	3	4	2	3	2	4	2	30
47	IS	B+	4	2	3	2	3	2	4	4	3	3	2	28
39	IS	B+	3	3	3	3	1	4	2	4	3	4	2	29
1001	OT	B+	5	5	4	4	3	4	3	3	2	5	2	35
40	MM	B	4	3	2	3	4	2	2	3	3	3	2	27
125	HW	B	3	2	4	2	2	2	2	2	4	3	2	25
75	MM	B	4	4	2	2	2	2	1	3	3	3	2	24
137	MM	B	4	4	3	4	2	4	3	2	2	5	2	31
129	MM	B	3	2	3	1	3	3	2	2	3	2	2	23
5	HW	B	4	3	1	1	1	1	2	2	3	4	1	19
56	HW	B	3	1	1	1	1	1	1	2	1	3	2	14
136	CM	B-	2	2	2	2	3	1	1	3	2	1	1	18
4	MM	B-	4	2	3	2	2	2	2	3	2	2	2	22
124	MM	B-	4	3	3	2	1	3	1	3	2	3	2	23
128	HW	B-	3	5	3	2	3	5	2	3	2	3	3	31
101	IS	B-	4	4	2	1	4	1	2	5	1	4	2	26
100	MM	B-	4	4	3	3	1	1	2	3	3	4	2	26
96	AS	B-	3	2	3	1	2	2	2	3	2	3	2	22
105	IS	B-	4	3	3	1	2	2	1	3	3	2	4	24
126	IS	C	3	2	3	1	2	2	1	4	4	2	2	23
42	IS	C	3	2	3	3	2	1	2	3	1	2	1	20
112	HW	C-	4	2	3	1	1	2	1	2	2	1	2	17
34	CM	C-	2	3	4	2	1	2	1	1	3	3	2	22
130	AS	C-	4	5	4	4	2	1	2	2	4	5	2	31
6	IS	C-	3	1	3	1	1	1	1	2	3	1	3	17
99	NW	C-	2	3	3	1	2	2	1	3	3	1	1	20
81	CM	C-	4	2	3	2	1	3	1	2	2	3	1	20
77	HW	C-	2	3	1	1	2	4	2	3	2	4	2	24
48	CM	C-	3	3	2	1	2	3	3	2	1	3	2	22
1002	MM	D	3	2	2	2	1	4	1	3	2	2	1	20
36	MM	D	3	4	2	2	1	2	2	1	2	3	1	20
71	HW	D	5	4	1	1	2	2	2	4	2	2	2	22
66	IS	E	3	4	2	1	1	1	1	1	3	3	1	18
117	IS	E	2	1	1	1	2	1	3	1	2	2	1	15

Figure 4: Projects assessed by development framework, ordered from top by grade. The darker shading indicates a 4 or 5 where this factor is used. Exemplar projects above the line.

Table 3: Comments from marking panel for C and fail groups

Wrote code as proof of concept prototype then tried to design it. 'one half of the team was designing and the other half coding and both at different rates'
Technical decisions made in getting prototype out haunted later development good process limited by technical skills
Functional prototype developed early but without design, that bad design stuck, Digitised a bad process as some stages of SDLC missed.
Initial concepts done with prototype, anything deemed to be too hard never got considered as a real requirement SDLC but skipped from initial idea to implementation
Couldn't achieve much technical complexity so scoped down to make prototype deliverable
After successful prototype almost no progress for the rest of the year. At last minute, they independently implement the rest of it and retrospectively did SDLC analysis
Prototype developed for proof of concept stuck SDLC very weak.. Client happy but insufficient technical complexity
Palm development. Never had stable platform. Couldn't make it work.

nise the potential bias of the authors in making this assessment, although this should have been partially mitigated by the impartial third person.

A simple summation of the eleven factors provides a very high correlation (r=0.756) with the grade received for each project (Figure 3: for this and subsequent statistics the exemplar projects are excluded). Further, the relationship is not affected by the type of project.

Figure 4 shows all the projects assessed according to the development framework. All factors are positively correlated with the grade.

Some anomalous projects can be identified: Project "78 HW A" was a hardware procurement project that did not follow the framework, whereas "174 IS A-" was a retrospective project where processes had been followed but evidence was lacking resulting in a slightly lower grade.

Most of the constituent factors that make up this score could be considered general measures of quality. Other measures such as efficacy of group work, or spelling, would probably also have positive relationships. This is still useful, we would also identify these factors as critical success factors. Some of the factors are not quite so obvious; the type of implementation – revolutionary or staged replacement as opposed to evolutionary – is striking in the pattern of being carried out by better groups.

Perhaps more important than the overall correlations of the process factors, are the relationships between them at different ends of the spectrum – what are the poorer groups doing differently?

By examining Figure 4 carefully (or looking at the correlations – not shown), some interesting patterns emerge. Poorer groups did less prototyping, which is understandable; they did less of other things too. What is critical is that when they did prototyping they did not also do "normal design" nor have stable systems, early delivery etc. Poorer groups that used prototypes in early stages also had very weak functional requirements, did not have strong testing plans etc, in other words the prototype became the development in total. Groups in the middle ranges that did prototypes tended to have weaker logical and physical design, instead they saw the development as one of making the prototype robust.

These patterns can also be seen in the comments of the marking panel for groups in the lower half of the grades (Table 3).

7. CONCLUSION

Hakim (2000) argued that the "challenge is to systematically incorporate prototypes into the software design and development lifecycle". In this paper we have identified a development framework which successfully does that integration. This framework was developed from exemplar projects and provides a best system framework.

Stein (2002) also identifies properties common to successful projects. But cautions "it is true however, for each property I can find … less successful project(s) containing the same property". By examining how the development framework compares with what poorer groups are doing, we have been able to identify how this framework might apply to weaker groups. In particular, we have demonstrated the risk of weaker groups doing prototypes instead of normal design, or conversely, sticking to a naïve view of the SDLC that results in poorly tested designs.

This paper has not considered the level of technical complexity of a project. Goold (2003) discusses the effect of both technical skills of group members and the scope of the proposed project. It would be worth examining the complexity of projects, and how this relates to both the grade and the development framework adopted. We found that the framework applies equally well across different types of project (see also Avrahani 2002).

The tension between product and process can be lessened by adopting a process that can be seen to produce good products while being flexible and robust. We believe that the development framework developed in this paper will provide a foundation for capstone courses.

Acknowledgements

We would like to thank the years of students for providing such a wealth of information. Also the work of the other lecturers on this course. Patricia Haden, Phoebe Eden-Mann and Zuzette van Vuuren helped with analysis. This research was carried out under Otago Polytechnic Category B Ethical Approval.

References

Barclay, K., Mann, S., Brook, P. and Doonan, A. (2001) *Development and Testing of an Adaptive Computer Keyboard Interface* Paper in the Proceedings of 14th Annual Conference of the National Advisory Committee on Computing Qualifications Napier, p13-22

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

Bridgeman, N. (2003). Project success: defining, designing, constructing and presenting a capstone project. 16th Annual NACCQ, Palmerston North, NACCQ.211-216

Clear, T. (2003). "The waterfall is dead, long live the waterfall!" SIGSCE Bulletin 35(4): 13-14.

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.

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

Fincher, S., M. Petre and M. Clark, Eds. (2001). Computer Science Project Work: Principles and Pragmatics. London, Springer.

Garrett, P., D. Youngman, J. McCormack, C. Rosescu and S. Mann (2003). Characteristics of success - virtually there. Proceedings of the 16th Annual NACCQ.269-272

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

Hakim, J. and T. Spitzer (2000). Effective prototyping for usability. *ACM Special Interest Group for Design of Communications*, Proceedings of IEEE professional communication society international professional communication conference and Proceedings of the 18th annual ACM international conference on Computer documentation: technology & teamwork.47-54

Hamilton, M. (2003). Familtrack. Proceedings of the 16th Annual NACCQ, NACCQ.484

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

Lawton, B. and A. Meikle (2003). Alternative CAD user interface. Proceedings of the 16th Annual NACCQ, NACCQ.487

Ponting, D., L. Quarrie and G. Robertson (2003). Serve i.t. right: hospitality training CD-rom. Proceedings of the 16th Annual NACCQ.495

Rudd, J., K. Stern and S. Isensee (1996). "Low vs High-fidelity debate." Interactions 3(1): 76-85.

Singh-Cosgrave, B., Sinclair-Fox,C., McLellan, G., Mann, S. and McGregor, G. (2001) *Interactive CD for Teaching Development Based Subjects* Concise Paper in the Proceedings of 14th Annual Conference of the National Advisory Committee on Computing Qualifications Napier, p460

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.