A+, Guaranteed[1]: Insisting on best of practice within capstone projects

[1] Conditions may apply

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

ABSTRACT

This paper provides a test of a best system development framework for undertaking capstone projects in ICT. A framework has previously been described that combines a software development life cycle approach with prototyping and early development. This framework is implemented with a cohort of 24 capstone projects. The paper describes the implications of the framework on the design of the capstone course and its implementation. The results show that project groups did follow the framework and significant increase in the average grade with a reduction in the failing grades. The relationships of parameters within the framework are examined. This shows that for weaker groups the incremental traditional SDLC process does not sit well with an early development model.

1. INTRODUCTION

Many ICT degrees include a capstone project (Fincher 2001). The choice of development process is pivotal in such projects (Beasley 2003; Chamillard and Braun 2002; Goold 2003;. Bridgeman 2003, Clear 2003). In computing education, the choice is complicated by the introduction of the teaching imperative. The issue we face is finding not only the best approach for development but the best approach for teaching it. Hakim (2000) argued that one "challenge is to systematically incorporate prototypes into the software design and development lifecycle". In previous work (Mann and Smith 2004) we identified a framework for undertaking capstone projects within an ICT degree, which, we argued, "successfully does that integration". This model emerged from an examination of exemplar projects. A high correlation was then found between the framework score and the marks received by a further set of existing projects. These projects had been undertaken with a direction of "use an approach that best suits your project".

In this paper we examine the effect of employing the framework in the design and management of subsequent capstone projects. This was undertaken with the intention of improving the outcomes for students and clients. The framework is described in the following section.

2. REVIEW

This best system framework was developed from exemplar projects:

1. "SDLC" identified methodology
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.

Mann and Smith (2004) used a simple "process score" that showed a strong correlation between the totality of the application of the framework and final project grade. This held across different project types. Although it was not reported in detail, while overall the different framework factors improved with the final grade, complex relationships seemed to emerge in lower grade groupings. Mann and Smith (2004) highlighted 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.

Mann and Smith (2004) were able to conclude that:

"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".

We hypothesise that implementing the framework will have positive effect on the final grades for students, and that this effect will be independent of project type. We also aim to further investigate the relationships between aspects of the framework.

3. METHOD

The framework was adopted for all capstone projects undertaken in the Bachelor of Information Technology at Otago Polytechnic in 2004. We describe the implementation of that framework and compare the results of the 2004 sample to those completed in 2001-2003 (Mann and Smith's exemplar and full sets).

Prior to the implementation of the framework, capstone groups developed their own methodology for the project. In 2004, the framework was made compulsory.

In implementing the framework the following instructions were given:

We require that you follow the SDLC under an umbrella of Agile Software Development. This gives the benefits of the structure of SDLC while adding flexibility of agile development.

The focus of the project is on the production of robust working systems (software, hardware and maintenance documentation). Planning, comprehensive development documentation and processes are important but are 'means to an end' with a focus on content rather than format/representation. It is expected that you discard most of the models you develop (although you do have to keep them for assessment!).

We also put emphasis on prototyping and testing. We are going through the SDLC twice. The first iteration is to analyse and design and release (usually to the client) a system that meets most of the functional requirements. The second iteration is intended to review the success of the first iteration in meeting business requirements, to review functional requirements (there will probably be more), and to deliver a stylish "bullet proof" implementation.

To enforce framework components such as early deliverables, the following milestones were set:

You are required to have at least four significant deliveries of system.

1. Requirements prototype. At the end of the analysis stage. This is a prototype aimed to test functional requirements.

2. Stable platform (week 12). The platform for your system should be developed and tested. For example, for a database with a web front end we would expect you to demonstrate connectivity and basic database functions via the web (insert, delete, query, update) plus any standard infrastructure (login etc).

3. Major release 1, week 25. Deliver to the client a system that meets most of their needs. This system should be usable and stable.

4. Major release 2, week 42. Deliver to the client a production system that meets all of their needs.

In order to reinforce the importance of delivering a functional system, the marking schedule template was altered as follows (previously was less structured but an example gave approximately equal weighting to the various stages of the SDLC):

Within the following bounds you write the marking schedule yourself.

Implementation 70%
Development Process 15-20%
Assessment documentation 5%
Critical review 5%
Learning outcomes 0-5%

Projects are undertaken by students on their own, or more usually, in groups of two or three. Groups are responsible for finding their client and project, although help is often given. Groups develop their own marking schedule based on the above template and present to a panel to justify the marks they have given themselves. This eight member panel consists mostly of industry representatives, the remainder being academics from the department. Some industry representatives have performed this role several times before, this having a moderating influence on the final grade. Project documentation, with confidential information removed, is lodged in the library, from which we extract comments and process information for this paper. The analysis of extent to which the projects followed the framework was carried out by an independent third party.

Table 1: Distribution of grades as quantity and percentage by group.

	A	B	C	D Fail	Total
Previous n	23	21	10	11	65
%	35.4	32.3	15	16.9	
Framework n	13	6	4	1	24
%	54.2	25	16	4.2	

Table 2: Comparisons of marks before and after framework imposed, by group and individual.

	N groups	Grade point average	Indiv n	Individualised grade point	% getting A
Previous	65	7.236	132	7.21	46
Framework	24	8.583	7.21	8.79	67

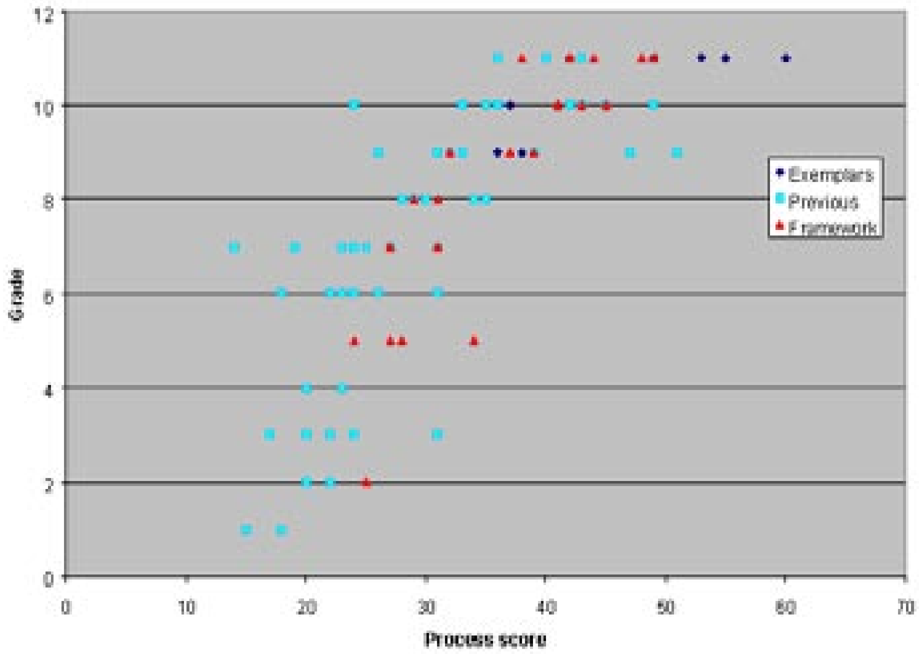

Figure 1: Relationship between process and grade point score before (exemplars and previous) and after imposition of framework.

For one project where the grade differed for different group members the average mark is used here. The mix of project types is unchanged from the prior sets.

4. RESULTS

Twenty four projects were undertaken in 2004. Table 1 shows the distribution of grades in comparison with the previous approach. This is a clear lifting of the grades from the previous projects.

The new average mark is for groups is 80.2% and when disaggregated back to individuals 80.29%. Unfortunately only the grades are available for the prior groups so only a grade point average system can be used in comparison (where D is 2, C- 3, C 4, and so on up to A+ being 11). Table 2 shows such a comparison which shows an improvement of the average of B to an A- (Table 2). There is also a very large increase in the percentage of individuals obtaining an A grade.

Can this significant improvement in grades be attributed to the imposition of the framework? There were no other changes in the process, the lecturers were the same, the mix of projects the

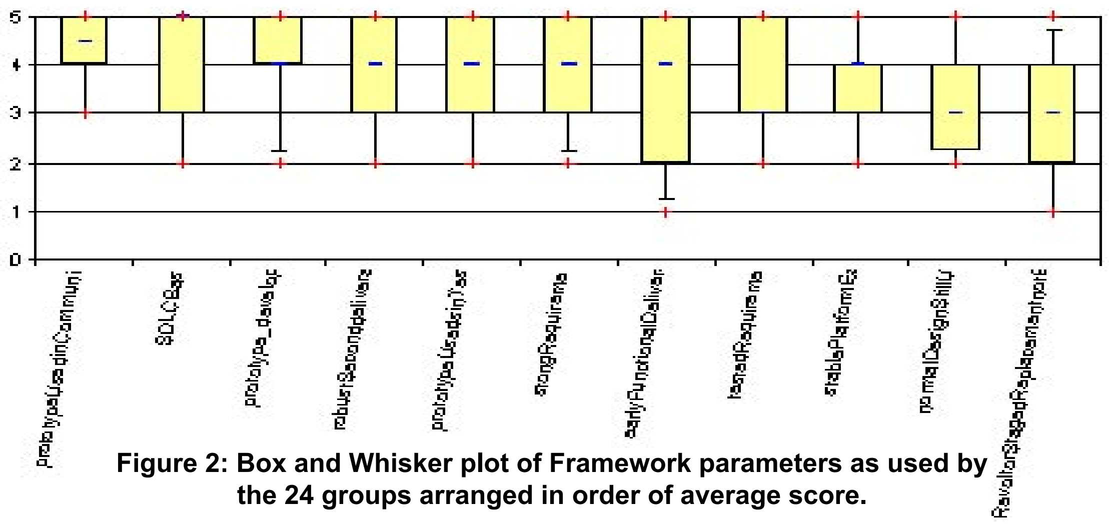

Figure 2: Box and Whisker plot of Framework parameters as used by the 24 groups arranged in order of average score.

same
tiona

Figure 1 shows the relationship between the process and grade point score before (exemplars and previous) and after imposition of the framework. When compared to the "Previous" set the Framework set shows a lifting of the overall grades. The assessed process score also shows an increase (shift to the right). This is to be expected, this score can be seen as an independently assessed compliance with the now required framework. There is a high correlation between the process score and the grade ($r^2 = 0.751828$). We have then a clear association between the framework and improved outcomes.

The next question involves the importance of the various components of the framework. Figure 2 shows that the parameters range from almost complete compliance (eg prototypes used in communication) through to parameters where 50% of groups only partially followed the framework (normal design, revolutionary rather than evolutionary development). The framework parameters with the highest degree of variability are the early functional deliverable, tested requirements and revolutionary development.

A multiple stepwise regression ($r^2 = 0.9497$) of framework components against the assessed mark selects strong requirements and revolutionary development as the most influential parameters, followed by the early development of a stable platform, the use of normal design and an early functional deliverable.

A factor analysis of framework components (Figure 3) provides two components that highlight interactions in the framework. These two

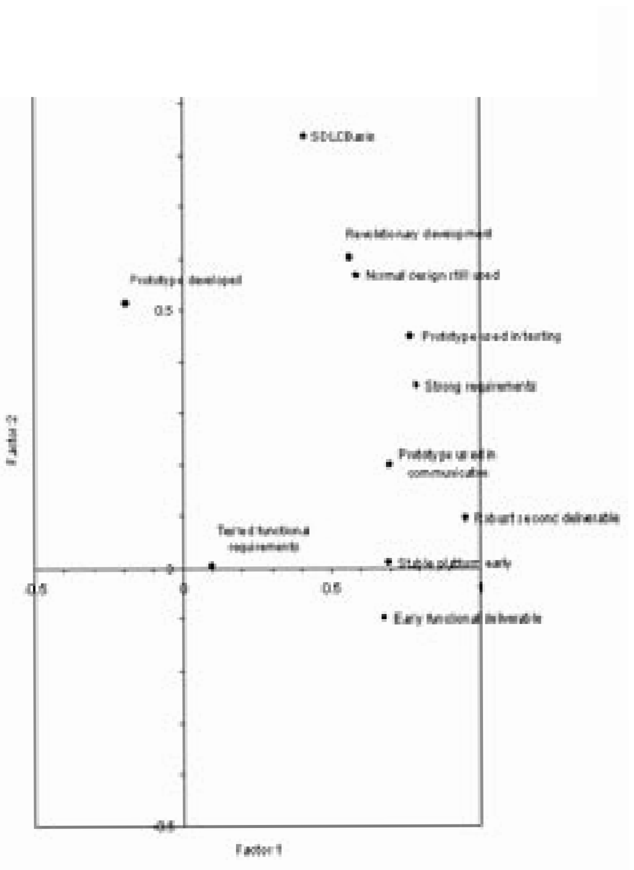

Figure 3: Factor analysis of framework components

factors account for 68.18% of total variation in the framework. The first factor is a continuum of prototype developed and tested functional requirements, against all other factors. When the factor scores for each group are then correlated against the final grade, this factor correlates to 65.12% (at 99% significance) of the final grade. The second component – early functional deliverable, use of a stable platform and tested functional requirements – act against the use of

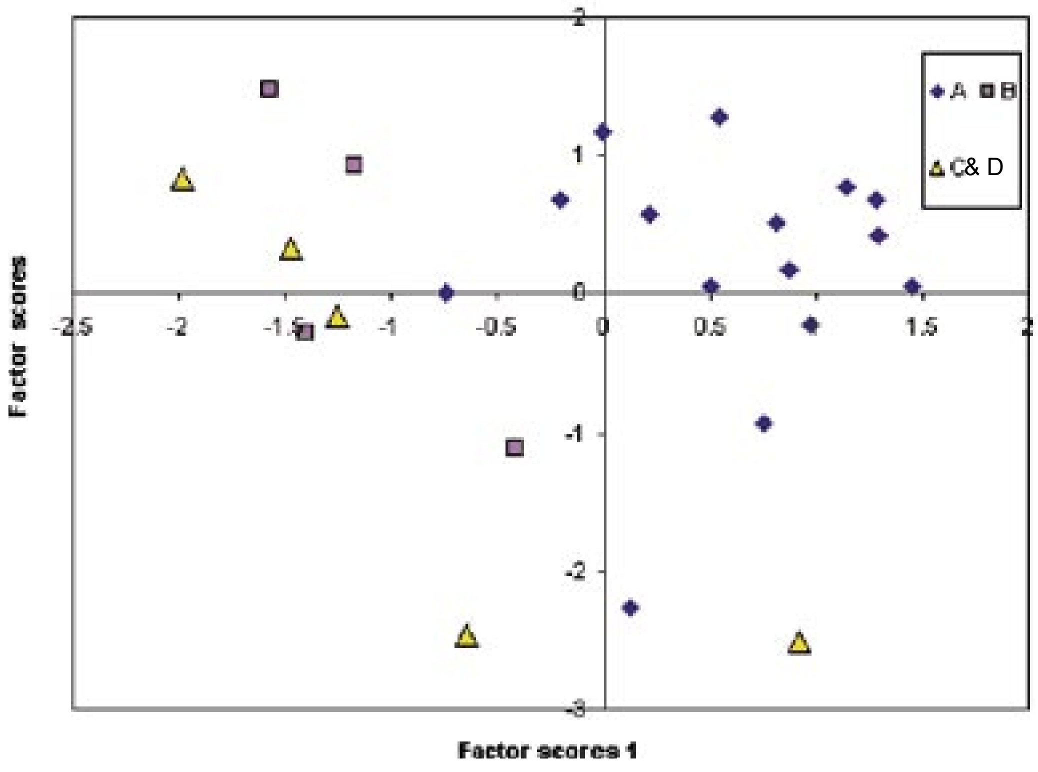

Figure 4: Final grades against factors, these two factors describe 83% of variation in marks. Axes determined by parameter distribution on Figure 3

the SDLC as the underlying basis. This factor, 18% of the variation in the framework parameters, correlates to 58.20% of the final grade. As Figure 4 shows, these two factors explain 83% (via regression) of the final mark. A third factor explained 9% of variable variation but did not significantly correlate with either the process score or the final mark.

These factors were unaffected by the type of project.

5 . DISCUSSION

These findings can be summarised as follows. The implications of these findings for teaching practice and further research are discussed in the next section.

1. The framework translates easily to implementation
2. Following the framework resulted in a significant improvement in final marks (average B to A-)
3. This shift occurs at both ends of the grade scale, the percentage getting A grade goes from 46% to 67%, the percentage getting C/D goes from 32% to 21%.
4. The extent to which the framework was followed varies across groups but this is unrelated to the type of project.
5. Weaker groups exhibit two patterns:
   5a Develop a prototype and initially test it, but this becomes the project, the subsequent robust development is poor
   5b Are poor at incorporating early development and testing into SDLC
6. The role of some framework areas require further investigation, especially the role of tested functional requirements.

Stein (2002) 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 applies to weaker groups.

Although based on a standard SDLC process (eg Hoffer *et al.* 2002), the implementation of the framework requires some shifts in thinking. The emphasis on prototyping and testing may be seen at to be at odds to a naïve view of the SDLC where no coding takes place until the implementation stage. Poorer groups may fall in to the trap of relying on the early prototypes as the basis for ongoing development rather than as tools to answer specific questions.

In an educational setting the early development (stable platform and early functional deliverable) has two advantages. First, it has inherent advantages in such an approach: a communication tool; client ownership of the solution, a robust product through onsite and production environment testing and so on. It also helps overcome the disadvantage of inexperienced developers. The students do not have experience either in development or the client's business area, nor a library of solutions to rely on. By encouraging them to develop code even while still otherwise in the analysis stage builds confidence that they will be able to implement the designed solution. The risk, however, is that the early solutions stick, bringing the danger of prototyping translating to the final system.

The role of tested functional requirements in the framework should be a subject of further research. This component is intended to ensure that functional requirements are verified as being sound requirements that both meet business requirements and are implementable. The early functional deliverable requires the group and the client to work through these functional requirements to identify a development plan taking into account both what is needed first and what is a sensible development order. Normally this comes down to a decision about whether to implement the core functionality (ie the hard bit) or the infrastructure (usually the easy bit). While the majority of groups do achieve this and go on to produce the rest of the SDLC and good final work, it seems to be a stumbling block for weaker groups. This uneasy relationship requires more work.

In Mann and Smith (2004) there was no examination of the effect of the ability of the students (ie A students get As). There we argued that the effect of this was inconsequential as we still wanted to know what it was they were doing despite their skills. We recognise that the effect of the grades of the students is a factor that should be examined here, and will be in future work.

Before we enforced the framework, students were free to identify their own development process. Anecdotally, some of the students described in this paper did not appreciate having a method imposed on them, despite it being described as a framework (rather than a rigid method), and the descriptions being heavily laced with agile terminology. In particular, groups undertaking hardware projects did not see the framework as being appropriate, despite it working for all project types in the exemplar, prior projects and now framework project sets. It would be worth qualitatively examining the reflective material contained in project documentation to examine such thematic material.

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, Boudewyn Erkins and Zuzette van Vuuren helped with analysis. This research was carried out under Otago Polytechnic Category B Ethical 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.

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.

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

Hoffer, J. A., J. F. George and J. S. Valacich (2002). *Modern Systems Analysis and Design*. Reading USA, Benjamin Cummings, 3rd Edition

Mann, S. and Smith, L.G. (2004) Role of the development methodology and prototyping within capstone projects. Proceedings 17th Annual NACCQ, Mann, S. & Clear, T. (eds). Christchurch. July 6-9th 2004. p119-128.

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

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.