Introduction of Software Hut Concepts into a Project-Based Course in Software Engineering

Jacob L. Cybulski
Department of Computer Science and Computer Engineering
La Trobe University

Abstract

This paper outlines a number of changes to the CS3PRJ unit of CS & CE degree at La Trobe University - the Software Engineering project (henceforth referred to as the *project*). The main benefits of the proposed modifications can be seen in the clarification of the project educational goals and methods, improvement of the assessment criteria, and finally in reduction of the project administration.

This paper first evaluates the current model of CS3PRJ, then proposes the new framework for running the project, and finally provides an example of a problem suitable for the new project structure.

The Current Model

Description. Currently the project is run in several stages reflecting a traditional software development life-cycle.

1. *Project planning (2 weeks, 0%).* Students are allocated to small programming *teams* of five to six persons responsible for the *project* development. Students will have an opportunity to nominate a particular team or project, although, it may not always be possible to meet all requests. Each project is allocated a *client* who is setting the project and who may be a member of the University staff or of an external company. In addition, each team will be allocated a *supervisor* who is normally a member of the University staff responsible for guiding student work on the project, in some cases the client and the supervisor may be the same person.

2. *Requirement statement (2 weeks, 5%).* Teams select their projects and after consultation with the client produce an informal statement of the project objectives.

3. *Requirement specification (5 weeks, 15%).* Teams refine and formalise their project requirements. Also, students propose the deliverables, schedules and the methodology they will follow during entire project development. A resulting specification document forms a contract between the client and the team.

4. *Preliminary design (3 weeks, 15%).* Preliminary design documents are derived from specifications formally accepted by the client.

5. *Detailed designs and implementation (13 weeks, 35%).* Students submit fully documented project deliverables throughout the rest of the academic year. All of the remaining project deadlines, the number and type of deliverable software modules, and their documentation depends on the previously agreed contract.

All stages are assessed by team supervisors. The final exam constitutes the remaining 30% of the subject marks.

Resources. In 1991, the CS3PRJ project work was supported by a series of thirteen (13) lectures and seven (7) case studies scheduled over the entire academic year. The CS3PRJ lecturer (1) and tutors (1) were responsible for organizing software development

teams, allocating projects, clients and supervisors, provision of guide-lines to supervisors, collection of marks, etc.

The 34 groups of five to six students (approximately 20 CSE2, 20 GDipCS and 150 CS3 part-time and full-time students) could select from amongst 12 projects, set by 13 clients, and controlled by 20 supervisors. The total number of University staff and outside people involved in the project was 25. On average, clients required 0.5 hour/week, and the supervisors 1 hour/week to oversee each of the team's progress (note that in most cases, the scheduled 0.5 hour/week of supervision was not realistic). In addition to an on-going team management supervisors also required approximately 1 hour to properly assess each of the project submissions (of which there is 4 pre-set deadlines, and 2-3 sub-system submissions each accompanied by a demonstration). The project coordination required on average 3 hours per week.

Running CS3PRJ over 26 teaching weeks necessitated approximately 20 hours of formal lectures (by the lecturer), 1326 hours of project supervision (by clients and supervisors), approximately 204 hours spent on project assessment (by supervisors), and very conservatively 78 hours of project coordination (by the project tutor). The total effort required to implement the practical component of CS3PRJ is estimated to be in the vicinity of 1608 man-hours.

Assessment. The Software Hut concept allowed CS3 students to greatly benefit from the project-directed course in Software Engineering, it enabled them to :-

- experience the real-life involvement in the planning and development of a medium to large scale software project;
- utilise a team work approach to the software development process;
- develop skills in relating to and negotiating with software clients;
- apply and integrate previously learned skills, develop new skills as appropriate.

At the same time the currently used model of the Software Hut suffers from the following problems :-

coordination

- too much overhead in running the project;

staff

- the project clients and the team supervisors frequently lacked familiarity with the objectives of Software Engineering;
- students' access to some of the clients and supervisors was frequently poor;
- several academic staff involved in the project were not committed to the subject nor the project they were supervising;
- project assessment as performed by the supervisors was frequently inadequate and late, thus, hampering the efforts to provide a meaningful feedback to students before they could start the next stage of their software development;
- external clients and supervisors always proved to be difficult to work with (for both the students and the academic staff);

students

- teams of students working on identical projects frequently colluded and copied both specifications and programs, there were no mechanism for easy detection of this practice nor the incentives to discourage it;
- in the past, specialisation of team members lead to several cases of students passing CS3PRJ with a high mark and no real understanding of the Software Engineering issues, addition of a formal exam was to eradicate the problem;
- the development effort of individual group members was hard to measure, and its distribution was frequently non-uniform within the group;

projects

- the projects offered to the development teams were of drastically different standard, some of the projects were too big, some too difficult or technical, others unacceptably easy;
- several projects were proposed by the clients in an ad-hoc manner, mainly serving the clients' research interests, rather than being guided by the educational objectives;

process

- requirement negotiation was often regarded as the most important part of the project, nevertheless, several teams were not offered the opportunity to negotiate their requirements, others suffered from the "super-realism" of their clients who proved to be overly difficult to deal with;

- projects and their assessment tended to be process- rather than product-oriented, this frequently lead to incomplete, incorrect, under- or over-specified projects;

- the real-life software life-cycle was not suitable for totally inexperienced software developers, from the initial stages of the development process students faced too big a task to correctly perform all the necessary steps;

- at no time, students were given any incentive (apart from the final mark) to produce good quality software and its documentation.

The Proposed Model

Description. The proposed project framework resembles more closely the original Horning and Wortman's model than that currently implemented in CS3PRJ. The CS3PRJ game centres around a *software hut*, an organisation responsible for commissioning software development, marketing and sales of software and services, keeping the registry of developed products and their licensing to development teams, resolving sub-contractors' grievances, etc. The Hut is directed by the CS3PRJ lecturer, it is run by representatives of student development teams under the supervision of a number of Software Engineering tutors. It consists of a number of business units governing the whole process :-

- business branch responsible for making software tenders, assessment of sub-contractors' proposals, offering and receiving the contract work;

- the finance branch responsible for the payment for the commissioned work, collecting license fees, paying the royalties from software sales, etc;

- software registry responsible for the purchase, copyrighting, and licensing of software products developed by the sub-contractors;

- the complaint department resolving all of the grievances between involved parties.

Similarly, to the present model, all of the students are allocated to a number of small software development teams (5 - 6 persons). Their role in the game is to act as software sub-contractors. The sub-contractors will be asked to perform a number of software development tasks, i.e. design and implementation of small software libraries, large product integration, and finally the specification, design implementation and maintenance of application programs. The entire development process will be organised into three stages of tool development, product integration and application programming, as follows :-

prelude

- all groups are formed and given a bank loan;

tool development (20%)

- the Hut is announcing a number of small software modules for tender;
- sub-contractors analyse the tenders and submit the proposals to the Hut;
- based of the received proposals, the Hut hands out the contracts;
- sub-contractors design, implement and document the software modules;
- the received software is assessed for its quality and potential price;
- the sub-contractors are paid for their products;

product integration (40%)

- the Hut commissions a new project which will be developed by all of the sub-contractors, and which will require modification and integration of several software units held in the Hut's software repository;

- the sub-contractors purchase all the necessary software tools (not containing the software units they've written previously) based on the price, functionality and the quality of its documentation, royalties from software sales are directed to the original authors;

- fully developed products are subsequently sold to the Hut, assessed, and paid for;

application development (40%)

- a final set of software applications is sent for tender by the Hut;

- the sub-contractors submit their proposals and are offered the projects, due to the nature of the courses taken by the team members, some of the teams may have no option of chosing their own applications (e.g. students majoring in accounting may have to select an accounting application);

- the teams purchase packages developed by their competitors, modify them to suit their specific needs, specify, design, implement and document the solution to the problem at hand;

- the applications are priced and purchased by the Hut;

finale

- as the final step in the process students are asked to issue dividends on their profits to all individual team members, the dividend level will be the foundation for the assessment in the project's practical component;

other sources of sub-contractors' income

- team representatives may choose to participate in the administration of the Software Hut, they may take part in the project assessment, teams may develop general-purpose software modules to be patented and subsequently sold to other development groups.

Note that at all stages of software development students will be asked to supply all the necessary documentation, i.e. module-level documentation for the tools, system-level documentation for the products, high-level requirements, specs and other documentation for the application programs.

Resources. The project will be supported by the same number of lectures and case studies as at this time. The supervision, however, will be considerably reduced. It is proposed that 1 lecturer and 4 part-time Software Engineering tutors with an average load of 10 hours/week will provide the necessary 1 hour consultations per each group (no client/supervisor distinction), and all the other necessary support for the CS3PRJ administration. It is suggested, however, that the tutors be involved in postgraduate work in the area of Software Engineering, and thus be fully committed to the field's principles, such an approach will also secure a few trained Software Engineering teachers for the future employment.

Project submissions were reduced from 6 to 3, also due to the students' participation in the project assessment, it is anticipated that tutors' marking effort to be reduced from 1 hour to 20 minutes per submission.

Assuming similar enrollment in CS3PRJ as in 1991, 34 teams will require 884 hours of direct supervision, 34 hours spent on the project assessment, certainly an increase in the project administration to 104 hours (assuming 1 hour per tutor per week), all totalling to 1022 hours, fully covered by the efforts of four half-tutors providing a total of 1040 man-hours (10 hours per week each over 26 teaching weeks).

We can observe a reduction of 568 man-hours, of which significant proportion was spent by highly qualified academic staff.

Assessment. The main objective of the proposed model of the Software Engineering project was to reduce the overhead in its management, increase the staff responsiveness to the students' needs, clarify educational objectives of the project, improve the standard of projects and their assessment, structure the project to be delivery rather than process oriented, while preserving the elements of software development methodology. The details follow :-

coordination

- the number of staff involved will be reduced from 25 down to 5, the number of projects from 12 to 1 (the last phase of the project allows for certain goal variations), coordination effort was reduced by 30%;

staff

- the project is run entirely by 4 half-tutors (including 1 head tutor who will take most of the administrative duties), the involvement of qualified academic staff was reduced who will be free to focus their interests on graduate projects;

- assuming that all of the 4 half-tutors to be involved in postgraduate study in Software Engineering (which is a preferable situation), all of the staff involved in CS3PRJ will be committed to the principles of Software Engineering;

- having a uniform tutorship scheme gives students better access to the staff running the project;

- project assessment will be built into the model thus a better assignment turn-around is expected - better feedback to students;
- CS3PRJ may provide some ground for supporting graduate research in Software Engineering (base funding of SE scholarships) and the training of future Software Engineering teachers;

students

- the sale/purchase model provides students with additional incentives to build good, well documented software;
- software patenting, and royalties from sales provides a mechanism for the reduction of illicit collusion and copying between students;
- students participation in the assessment process, gives them a better understanding of the software quality;

projects

- better control of the project standards (1 project with variations);
- the project may be designed with clear educational objectives;

process

- the negotiation process is less formal and may be done on the team-supervisor level, the client's role has been eliminated;
- the project's emphasis has been shifted from process to product orientation, while preserving the need for methodology and quality documentation;
- the model inverts the software life-cycle to allow students to progress from simple and familiar (program design) to complex and novel (integration and application), while saving all of the life-cycle components, but at the same time reducing the complexity of any particular stage;
- the model emphasizes the students' ability to work with foreign code, code reuse, and its maintenance.

A Sample Project

A SPREADSHEET project may be used as a model for the CS3PRJ :-

1. *Tool development.* Several groups are asked to develop the following modules: a simple form-based display mechanism, a formula parser, a simple text editor, a sparse matrix, a calculator, and a filing system.

2. *Product integration.* All groups are asked to produce a spreadsheet calculator, which would allow visual editing of data and formulae, saving and retrieving of data into and from a file.

3. *Application programming.* Students are asked to use a spreadsheet calculator to construct the following programs: either a tax calculator, a general ledger, a simple stock inventory system, a CPM/PERT model, etc.

Conclusions & Implementation

This paper discussed the details of the current model of CS3PRJ and proposed an improved framework for the subject practical component. The main benefit of the new approach is a reduction in effort of the supervisory staff by 30% (especially that of academic members of staff), in improvement of the subject educational objectives, and provision of additional incentives to students. The proposed model will require 4 half-tutors enrolled as graduate students in Software Engineering, each providing 10 hours/week of their time.

The new model of Software Engineering teaching will be introduced in two stages. In the first year it is planned to test the feasibility of running the project with four support staff only (i.e. 4 half-tutors). The model will be simplified to eliminate the project swapping (which would inject attract additional supervision effort), and to structure the life-cycle in a traditional manner (to preserve elements of the previous structure already in place). In the second year a fully featured model of the Software Hut will be implemented.

Bibliography

J.J. Horning and D.B. Wortman. Software hut: a computer program engineering project in a form of a game. *IEEE Trans. Soft. Eng.* 3(4):325-330, 1977.

D.B. Wortman. Software projects in an academic environment. *IEEE Trans. Soft. Eng.* 13(11):1176-1181, 1987.

V.B. Ciesielski, K. Reed and J.L. Cybulski, Experience with a project oriented course in Software Engineering. *Proc. Australian Soft. Eng. Conf.*, Canberra, NSW, May 1988.