Transactions on Information and Communications Technologies vol 7, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

Software engineering: from theory to practice

M. Merabti, C. Bamford
School of Computing and Maths, Liverpool John Moores University, Liverpool, UK

Abstract

One of the topics missing from many of today's Computer Science/Software Engineering courses is a course where the students would get three types of input: (1) a substantial project before the final year (usually only small revision questions or exercises are offered); (2) a programme of study that binds all learning objectives within a particular level and (3) a preparation for eventual employment by for example working in teams for problem solving, organisational, and practical management skills. In this paper we will describe our experience in running one such course entitled: "An Applications Workshop Programme". We shall detail the philosophy behind the design, the organisation of the programme, the applications to be undertaken by the students, the organisation of the work between the students, and the lessons learned from running this series of workshops over all levels, but in particular the second year.

1 Introduction

The software crisis has arisen in the continuing drive for ever more sophisticated software which has led to larger and more complex systems. The aims of quality and reliability in the systems have required the introduction of engineering principles into the whole design and development process. Software engineering is a large subject area, combining mathematical principles for the theoretical modelling of the systems with the sound practical methods and procedures for development.

With the software industry needing to use such principles, they were introduced into higher educational courses, initially as part of more general Computer Science courses, but lately as specialist courses in Software Engineering.

The limited time available on a higher education course has provided a challenge for curriculum designers who have to select and package the material to provide suitable learning objectives for the students. In [1],

Finkelstein presents a guide to and comparative analysis of curricula in certain computing courses in Europe, while in the UK, curriculum guidance for software engineering has been published by BCS/IEE joint working party [2].

As Bott [3] points out, it is not possible for software engineering courses to produce fully qualified software engineers. They should provide the educational background for an additional substantial period of professional training and experience. One area in particular which needs reinforcement and development is the practical application of the methods and procedures learned in the more theoretical modules to larger-scale problems. Such problems necessarily involve group work, project planning and management, interaction and cooperation of group members. Dawson et al [4] describe a short professional training course used at GEC-Plessey Telecommunications to introduce new software engineering graduates to the "real world" environment. It is a project-based course designed to introduce the graduates to the working methods and practices of the company and to involve them in group work.

In the design of the BSc Software Engineering route at Liverpool John Moores University (LJMU), which is a modular scheme sharing some of its modules with other degree routes, we included one module at each level of the scheme to form an Applications Workshop programme. This paper describes the philosophy behind its design, its organisation, the kind of projects undertaken by the students and the lessons learned from running the programme.

2 Degree route structure

At LJMU, the degree courses in computing are run in a modular scheme. Each of the degree awards has a defined set of modules which the students on that scheme must take (Core modules) together with others that the students choose to take (Options/Electives). The BSc routes are run with modules at three levels, with students on Sandwich courses taking levels 1 and 2 in the first two years of study, and level 3 in the fourth year after a year on an industrial placement.

The modular degree scheme has been running in a trimester mode with individual modules running over one or two trimesters. Each module has a credit rating indicating the total amount of student learning time allocated to it. This includes lectures, tutorials, practical sessions and private study activity. Typically the modules have credit ratings of 2 or 3 allowing for 60 or 90 hours of learning activity. (Each credit has an allocation of 30 hours of learning activity.) One exception is a final year project of 6 credits. In this module the student undertakes a substantial individual project, starting in the first trimester and submitting the final project report early in the third trimester.

Although the project has undoubted value in providing the student with the opportunity to undertake a more substantial practical task, using many of the principles and methods learned elsewhere on the route, it has limitations:

- it is an individual project so does not involve group work.
- it has to be achievable within the time scale by a single student, so is not a large-scale problem.
- no equivalent modules exist at levels 1 and 2.

For the Software Engineering route, it was decided to retain the project in the final year but to supplement it with modules at each level requiring larger-scale problems to be tackled by the students working in groups. This was called the Application Workshop programme. They were designated as core modules for the route and available as options/electives to students on other computing routes.

3 Applications workshop programme

The workshop programme consists of three modules, one at each level: Level A Programming Applications Workshop (level 1), A Software Engineering Applications (level 2), and a Software Engineering Development Workshop at level 3.

The level 1 and 3 modules were allocated 3 credits each, while the level 2 module was allocated 4 credits. In a recent revision of the modular scheme, all have been reassessed at 3 credits each.

The design of the workshop modules at each level of the route had to take into account which modules the student would have already completed. In the level 1 module, offered in the last trimester of the year, the students would have completed introductory modules in programming and program design. To revise and reinforce this knowledge, the module begins with a series of small workshop exercises which then develop into larger mini-projects in which the students have to integrate their own program code into existing programs. The need for documentation and testing is emphasised during the workshop.

By the time the level 2 workshop module is taken, the students have met more advanced topics in programming, data structures, object-oriented programming, software management and are studying formalism in software specification. The workshop is organised around individual and group exercises designed to illustrate and integrate the material met in these other modules. The problems examined here are larger than students have experienced before and require cooperative group work. The students are encouraged to employ the principles and techniques learned in the course to the sizing, planning, management and implementation of a group project.

The level 3 workshop module is based around a case study in which the students work in groups on various aspects of a large scale problem. They have to follow the project through its full life cycle.

3.1 Workshop Organisation

Although much of the organisation, requirements, and philosophy are common to all levels of the workshops programme the rest of this paper will concentrate on the level two applications workshop. We believe that the impact oft his practical course is most beneficial at this level.

Group selection Although it has been suggested that for a group to be effective it needs to consist of six to eight students [5], our groups have varied from eight to eleven members. This depended mainly on the class size. Group members are either chosen randomly or alphabetically by the project coordinator with no input from the students. This decision was taken to reflect two main reasons: (1) employees in industry do not always have the choice of who to work with and (2) our experience in running other group exercises where students were given a free hand in constituting their own groups as well

as drawing from other studies [6]. The latter option has resulted in cases of able students getting together and therefore skewing the effect of the learning experience in the class. Another problem has been some students who do not mix very well with their class mates being left out on their own. However, imposing group structures, although resulting in a good mix ability groups, could result in resentment in some cases. To get around this problem, such groups have benefited from counselling and closer supervision throughout to ensure that there are not too many problems of this nature per group. In addition, our marking scheme, discussed later, took into account of these problems to ensure that these problem groups were not penalised.

Group organisation The group organisation has been left entirely to the discretion of the group. Groups have varied in their management, with some appointing a chief programmer team structure [7] and others were run democratically. These decisions tended to vary with the groups' individuals and in some cases, an individual playing a certain role has been replaced by another team member or some of their work load has been split among other team members at a later date.

Groups have split tasks as they saw fit. Some included responsibility for design, documentation, programming as main responsibilities or have organised themselves as subteams for all tasks.

To avoid too much disparity in the work undertaken by groups members, teams were strongly encouraged to involve more team members in most of the tasks. This has resulted in not penalising the weaker students for example, by not involving them in design or programming stages.

Project selection In some less substantial projects and exercises that we have run in other courses, and in the first year Applications Programming Workshop, groups were given a variety of options such as each group has its own project which is different from the rest of the groups. This method has also been attempted by other teachers elsewhere [8]. For this rather substantial piece of work (4 credits and running over an entire semester) we have decided that ali groups would attempt the same project. A number of benefits accrue from this decision. It makes the assessment more objective as it facilitates comparison between the progress of the different groups. It provides a healthy dose of competition between the different groups. It is fair to the students, and it reduces the usual complaints of others' project being easier than their own. The last point is particularly pertinent as the students have no input as to which project they should undertake.

Project scope The aim of the CS 241 course (Applications Workshop) is

- To introduce the student to the problems of building complex software systems.
- To promote the development of self management skills.
- To develop practical design skills.
- To develop group working and presentation skills.
- To use software development tools appropriately

A group of around eight members is fairly similar to many small scale software project groups within industry, thus the course mimics many of the features of actual project development. The principle behind the project is that

it demonstrates many of the problems of software engineering work. As a result the main problems faced with the project is that the problem is difficult to understand and develop. The problem to be addressed by the group is :-

- Complex with many possible solutions
- Open ended in that students have to decide when they feel they have solved it.
- Under-specified in that it is not clear how soluble the problem is.
- Involves the mastering of new techniques and skills with limited formal training.

Each of these features is a key part of the problem and many of the frustrations they feel are those that most engineers feel when undertaking any complex task. Students will have regular design sessions where they coordinate their work with the supervisor (lecturer).

However, it is the students who manage the approach to the problem not the lecturer(s). During the course, they decide the work that is needed. How it is to be built. And they will have to determine how to apply the knowledge they have learned across the second year course to the problem.

3.2 A case study

The topic for students on CS 241 is to develop an object oriented design toolkit. This tool kit should support parts of the design process using separate tools that can communicate with each other. These tools should include:-

- A design entity dictionary which provides facilities to locate and display entities and can support different versions of entities.
- An object communication tool that allows object communication diagrams to be drawn.
- A code generator tool which takes the output from the communication tool in addition to output from object definition tool to generate template C++ reflecting the object.
- An object repository which stores developed objects with the associated design descriptions and code which realises the object. This repository should provide an external interface for a number of tools to use.
- A browser for the object repository which allows users to interrogate the object repository to discover previous objects which could be reused.

These need to be supported by a tool communication infrastructure. This tool communication infrastructure should reflect a particular software architecture. Students start by choosing a particular architecture for the toolset and choose an appropriate communication mechanism.

The initial phase of the project involves them working out a better definition of the problem and deciding how they wish to go about solving it.

3.3 Supervision

Members of staff are available during the project to provide support and guidance. Each group is allocated a supervisor with whom the students will hold regular (weekly) coordination meetings. In these meetings design decisions and other issues as well as problems are discussed. In addition,

other meetings are arranged with other members of staff regarding specific problems. For example, the use of a given formal specification language or programming environment. The overall coordination is undertaken by a lecturer in charge of the programme The responsibilities include group selections, coordination with lecturers, coordination with students, the allocation of marks across the groups, and the presentation of marks to the exam board.

Resources and budget The resource allocation to the project has been greatly facilitated by the fact that the group members are drawn from mainly the Software Engineering degree course. These students, and the few other possible candidates, have all benefited from a number of common learning experiences. They are all familiar with the school's possible platforms for development, namely, Microsoft Windows, and Unix workstations using X windows with motif. All of our computer related degree students are familiar with the C language before this project is undertaken. In addition, SE students, have also been introduced to Object Oriented Programming and C++. The familiarisation with the C programming language was found to be very useful for the placement year and the final year project which is undertaken by all our degree students.

3.4 Assessment

The project is intended to encourage students to develop self management and planning skills and we expect the students to develop their own timetable of work. Many of the reports to be handed in, initially, require the development of plans of action to realise the designs. The first deliverable is the development of a plan for completing the project. After that we expect students to be as much as possible self organising and to submit short individual project reports throughout the duration of the project culminating with a final group report.

Project plan A detailed workplan for the whole project showing an allocation of work across the group and a scheduling of deliverables to support the development of a complete toolset. This deliverable also needs to provide information on:-

- How the project configuration will be managed
- What different procedures will be used to support design and code development.
- When important milestones will be reached and how they will be assessed.
- How different parts of the project will keep in step.

Weekly reports Once the detailed workplan had been agreed the students are expected to report weekly by electronic mail to the project coordinator about the progress of the individual pieces of work they are undertaking. This arrangement has three functions: (1) make sure that the students keep a diary of their work, which is useful for writing the final report (2) a deliverable that is used for allocating individual marks at the end of the project, and (3) as an important encouragement (pressure) to the individual student to produce some work every week. In addition, a weekly meeting reporting how the group is performing as a whole is held with the project coordinator and the groups

spokesperson. In these meeting any problems, in particular non-technical ones are aired, and possible solutions are given. The spokesperson is democratically chosen by the group and is usually different from the chief programmer where the group has decided on this type of structure for the management of the work. At times, when some technical problems or otherwise have seemed to be too difficult to be solved satisfactorily among the group members, even after involving the assigned group tutor/lecturer, a group meeting of the spokesperson and chief programmer is organised with the project coordinator.

Project report The final deliverable is the project report. This report includes the requirements specification, the design decisions, the implementation paths and their description, and a user manual on how to use the environment. The students are also encouraged, in this exercise, to make sure that all relevant sources that influenced their design decisions, such as published papers and books, are referred to in the document. For each of these tasks, students are free to use the appropriate method or tool. For example, some students use entity relationships diagrams to represent the requirements, others use formal specification languages.

Presentation Part of the assessment at the end of the module is for each group to do a presentation of the achieved work. This presentation usually takes the form of a live demonstration of the software and an explanation of all the different parts. The demonstration is usually performed by two group members who are familiar with the totality of the tasks. This team usually consists of the spokesperson and chief programmer when there is one. This demonstration also serves to show how well the initial objectives have been attained.

Marks allocation The marking scheme has been one of the hardest task to perform. We needed to devise a scheme where students are given the marks that they have truly earned. This is particularly difficult to achieve for the individual member of the group. To this end we use three sources for the eventual mark: the group mark allocated for the overall deliverable one, deliverable two and presentation; the individual mark, given to the weekly reports; and a mark for effort and contribution. The effort and contribution mark is a percentage that the members feel a particular person has contributed. For example, in a group of eight, each individual would allocate to the rest of the team what they think constitutes their share of the group's mark, for example 80% or 40%. We found this to be quite successful in catching the people that do not do any work and therefore hope to benefit from the work of others. The group's mark is allocated by the coordinator and the lecturers involved, based on the deliverables and presentation. The individual mark is worked out using the following algorithm: percentage of group mark (using weekly report to confirm that assessment is fair) and add or delete 10% or leave alone depending on the assessed individual contribution to the project as a whole as performed by the lecturers and coordinator through the weekly report, presentation and final report. This mark is allocated for leadership, additional management skills, importance of bonding the group together and achieving ultimate goals. Negative effects or non contributions can be penalised further by reducing the mark by 10%.

4 Conclusion

The applications workshop programme has been received with a lot of enthusiasm by staff and students alike. The choice of applications has been found to be very attractive to students so much so that some of them are trying to complete it in their own time in their industrial placement year. Students, up to this project, have not been used to work with under-specified and very large projects and therefore have some apprehension which necessitate close initial supervision. However, even these difficult times are appreciated by the students as necessary towards the end of the project. Although, realistically, groups are not expected to deliver completed prototypes many of them have complete and detailed design with many of the parts implemented and operational. Undertaking to run a course, like this one, does necessitate a commitment and enthusiasm from a number of lecturers, in addition, to the coordinator. The processing of weekly electronic reports represents a mammoth task. We are at present investigating the creation of some generic electronic form to facilitate the processing of these reports.

Acknowledgements

Thanks are due to T. Rodden and B. El Galal (Both of Lancaster University) for a fruitful exchange of ideas in the beginning of the project. Thanks are also due to LJMU colleagues have helped run the course, in particular, John Willits and Keith Whitely for a particular input to the project management side of the course and the overall supervision.

5 References

1. Finkelstein, A., European Computing Curricula: A Guide and Comparative Analysis, *The Computer Journal*, 1993, **36**(4), 299-319.

2. BCS/IEE, *A report on undergraduate curricula for software engineering*, Institution of Electrical Engineers, London, 1989.

3. Bott, F., Software Engineering Education, *Software Engineering Journal*, 1989, **4**(4), 174-176.

4. Dawson R.J,. Newsham, R.W., & Kerridge, R.S., Introducing new software engineering graduates to the 'real world' at the GPT company, *Software Engineering Journal*, May 1992, 171-176.

5. Mumford, E., "Helping people design their own jobs", *I/S Anal.*, 1987, **25**(12), pp. 13-14.

6. Oman, P., Software Engineering Practicums, *SIGCSE Bull.*, 1986, **18**(2), pp. 53-57.

7. Sommerville, I., *Software Engineering*, 4th edition, Addison-Wesley, 1992.

8. Bullard, C.I., Caldwell, I., Harrel, J., Hinkle, C., & Offutt, A., Anatomy of a software engineering project, *SIGCSE Bull.*, **20**(1), 1988, pp. 129-1.