Group Projects in Software Engineering at York

Dr Jim Briggs

Department of Computer Science
University of York
York, YO1 5DD, U.K.
jim@minster.york.ac.uk

ABSTRACT

This paper describes the format of the second year group project in software engineering undertaken by all single-subject undergraduate students in Computer Science at the University of York.

Introduction

We have been running group projects in software engineering at York since 1979, though in 1979 the group consisted of only one student! Currently the group project is a core course unit for all students on our MEng Computer Systems and Software Engineering and BSc/BEng Computer Science degree courses. The project is also an optional unit for a small number of Norwegian students visiting the Department on an exchange scheme from the University of Oslo. The only undergraduate students who do not do the project are those on our combined degree course with mathematics, but it is our aim and intention to involve those students in the near future. The total number of students doing the project is 60-80 per year.

A number of students taking the course unit have some prior industrial experience. The MEng students and a small number of the BSc/BEng students have undertaken at least one vacation placement with their sponsoring company. Increasingly we have a number of mature students on our courses, many of whom have worked in the computing industry before commencing their studies. Typically the number of students in all these categories amounts to 25% of the total.

The project fits into the curriculum in the summer term of the second year. The students therefore have had the conventional first year course units in introductory programming, algorithms and data structures, supplemented by a lecture course on the formal construction of programs, well before they come to do the project. In the term immediately preceding the project they are given an 18 lecture unit on System Specification that acts as a theoretical companion course. The System Specification unit reviews a variety of system specification techniques and studies the requirements methodology CORE[1] and the specification language Z[2] in detail. Students are introduced to the problems of large scale software development and the software life-cycle, and they learn systematic approaches to the specification of requirements and system architecture, and spend some time considering management techniques.

Objectives

The primary objective of the group project is to give the students some experience of working together in groups. Although it is not the first or only opportunity that the students have for not working alone, it is the only course unit where they work together in large groups. Very much secondary in our view is the aim to teach the students new techniques in software engineering. As was stated earlier, by the time they get to the stage of the project they have already had several lecture courses to teach them various aspects of the topic. The project however does reinforce much of what they have learned in theory, for example it forces them to practice the use of CORE and Z taught to them in the previous term. Indeed the aim is to expose them to the problems of team working to make them realise how the techniques facilitate communication and decision making in a team.

Format

The class of 60-80 students is divided into 4 teams (each of 15 to 20) and each team is divided into 5 groups, each of 3 or 4 students. Each of the 4 teams is working on the same problem, but each team produces its system independently of the work of the others. Within each team, each group is responsible for the production of part of its team's system.

The production of a system consists of the following stages:

(i) production of a CORE specification of the functionality of the sub-components of the system;

(ii) formal specification of parts of the system in Z;

(iii) implementation (in Ada);

(iv) documentation;

(v) testing (independently by groups, and in combination with other groups).

Each group implements its specifications, tests them, and combines their work with others in their team to produce a working system.

This year the problem set was one to implement ATM software for a mythical bank. The five groups within each team were appointed responsible for the bank staff's user interface, the customer user interface, the accounts database, the transactions database and the updating of accounts. Other recent problems have included an examination marks processing system, software for a mythical television company's parliamentary

election coverage, a system for pricing menus in a cafeteria, a departmental timetabling system, a literate programming system, a program testbed generator and a communication system for the Roman army.

Assessment

The project is assessed in four parts, two (deliverables A and B) corresponding to material taught in the System Specification unit, and two (deliverables C and D) corresponding to the project itself.

For deliverable A each team submits a CORE specification for the whole system, and each group submits a Z specification for their subsystem. The specifications must be submitted in the fifth week of the project.

For deliverable B each student produces an individual report on their specifications. In his/her report, the student must identify the major early design decisions implicit in the CORE specification for which the student's group was responsible, and explain why these, rather than any other, decisions were appropriate. In addition each student must give an informal description of the mapping from the CORE specification to the Z specification, and from the Z specification to the Ada source code, of the parts of the system for which the student was individually responsible. Credit is given for identifying the key issues in the specification and implementation and explaining why the most critical design decisions were made. This report is submitted in the seventh week of the project.

At the end of the ninth (and last) week the final two deliverables are submitted. For deliverable C each team submits their:

(i) complete specification of the problem;

(ii) program listing;

(iii) documentation;

(iv) test results;

(v) costing.

The project costing should indicate how much the system has cost to produce, i.e. the real cost of the equipment, staff and student time expended on the project. The students are given various figures relating to the cost of the computing facilities they use, and told to assume a proportion of their student grant as labour costs. Staff costs are in terms of the number of hours of lecturer or demonstrator time spent supervising the project.

For deliverable D each student submits an individual report on the project as a whole. This report is in two main parts:

(i) a summary of the work done, in the form of a technical diary arranged chronologically;

(ii) advice offered to someone wishing to start a similar project the following year, addressing in particular any two of the following topics:

- the usefulness of the specification towards the development of the system;
- the usefulness of the textbooks which he/she consulted;
- the implemented part of the system and the results of testing;
- the effectiveness of the management structure of his/her team and/or group.

The emphasis is expected to be on the specification, design, testing and management aspects of the project. Perceptive comments about software engineering, working in groups, using specifications, etc. receive the most credit.

The marking of the deliverables concentrates primarily on the individual reports, and uses the team deliverables largely as reference material. Instead of assessing the students on what they produce as a result of the project, we are in fact assessing them on their personal review of what they did, how they did it and, most importantly, why they did it that way. This avoids the problem of either giving all students working together the same mark, or of separating out each's contribution and assessing it separately. It also gives scope for the student who does not contribute as actively to the product as the others to write a report justifying his lack of contribution. Though unlikely, it is conceivable that a reasoned argument as to why the student was best kept away from the work at hand could attract as good a mark as any other!

Staff resources

Staff time devoted to the administration and supervision of the group projects is organised so that the peaks of effort occur during vacations and the troughs occur during the intense examination-paper marking period in June. Two days in March are spent writing the project specification. The major part of this is developing a scenario into which the project can fit; the detail of the specification, including the instructions to the students concerning format, assessment and timetabling varies little from one year to the next.

During the course of the project, staff time is limited to three contact hours per week. These form a contiguous timetabled period during which the student teams are expected to meet, either to carry out their work or to plan how to do it. A single member of the lecturing staff supervises the four teams, wandering between the four adjacent teaching rooms allocated to them, observing the students, answering questions on the specification, and, occasionally, offering advice. The member of staff is supported by a graduate demonstrator who performs a similar supervisory role, and who can offer technical advice on CORE, Z, etc.

The biggest chunk of staff time is consumed after the project is completed. Spending an estimated 20 minutes on each student's report means that something like 40 hours in total are devoted to assessment. This normally takes place during the summer vacation. The marks are carried forward and included as part of the third year assessment scheme.

Student workload

From the students' point of view, the project takes place over a period of nine weeks. For the first six weeks, they are taking four other course units in parallel with the project. In that period, students are expected to spend a notional six hours per week on the project, of which three hours are timetabled. Time will be spent in meetings with group colleagues, management

meetings according to their team's adopted structure, and working alone on specifications and code. During the last three weeks of the term there are no timetabled lectures, so the students are expected to spend as much time as is necessary to get the project finished. Typically, a student will spend most of this period writing or testing code, and integrating theirs with that of others. In the closing stages, everyone will go off by themselves to write their individual reports.

The following is given to the students as a rough guide to the progress they should be making on the project:

Week 1 Introduction to project and groups. Discussion. Initial ideas for CORE diagrams.

Week 2 Finalising CORE diagrams. Start of formal specification.

Week 3 Continuation of formal specification.

Week 4 Completion of formal specification of the system.

Week 5 Submission of specifications. Design program.

Week 6 Implementation.

Week 7 Submission of specification report. Implementation and testing.

Week 8 Implementation and testing.

Week 9 Submit program and project report.

The students have unrestricted access to the department's computing facilities (an Orion 1/07 is dedicated to undergraduate use). Terminals are available 24 hours a day, seven days a week. As might be expected, teams make little use of the computer during the early specification stages, though some teams do set up elaborate intra-team communication mechanisms. During the coding, testing and integration phases, students can use any software development tools they wish – either standard UNIX tools or ones of their own.

It is interesting each year to observe the management structures that the student teams set up. Most teams appoint an individual to head the team, though they differ in whether he (never yet a she) is an appointed "leader" or an elected "chairman". Usually groups within each team appoint a group leader, and often the five group leaders will form a sort of board of directors to take higher-level decisions within the project or to provide a means of communication and co-ordination between groups. Some teams meet together in their entirety while some groups never speak to other groups except through "formal" channels. It must be said that the most successful groups in recent years have tended to be those that are strongly led by a leader with solid technical skills and, invariably, some previous industrial experience.

Periodically, each team is asked to give a ten minute presentation on some aspect of their work, for example their progress during the past week and plan of action for the following week, or the external characteristics of the user interface to the system. These presentations are made to the lecturer and demonstrator (but not to the other teams) and take place during the timetabled weekly session, typically at only one hour's notice to the students. This gives the opportunity for the staff to find out what is going on in the teams and to comment on the technical decisions that the students have made. For the students it gives them an opportunity to focus or review their strategy and gives one or more of them per team the opportunity to stand up and speak to a room of 15-20 people. These presentations are not assessed.

Conclusion

We have run group projects in software engineering at York for several years as an exercise to give the students experience of working together rather than individually. The setting up and administration of the project is fairly straight-forward and, except for the marking, takes up little staff time. Overwhelmingly the students seem to enjoy the project, and their potential employers often show interest in their experience of it at interview. The professional bodies that accredit our courses (the British Computer Society and the Institution of Electrical Engineers) express satisfaction with it. The bottom line is that, by doing the project, students learn that software engineering is difficult, and that working in groups and as a team is difficult.

References

1. G.P. Mullery, "CORE - A Method for Controlled Requirement Specification", *IEEE Computer Society Press*, Proc 4th International Conference on Software Engineering (1979).
2. J.M. Spivey, *The Z notation: a reference manual*, Prentice Hall (1989).

****

AAPT-- continued from page 47

Stasko J. T., (1990), TANGO, A Framework and System for Algorithm Animation. *IEEE Computer*, Vol 23, No 9, pp 27 - 39.

Stone D., (1989), Using Cumulative Graphic Traces in the Visualisation of Sorting Algorithms., *ACM SIGCSE Bulletin*, Vol. 21, No. 4, pp 37 - 42.

Waguespack L. J., (1989), Visual Metaphors for Teaching Programming Concepts. *ACM SIGCSE Bulletin*, Vol. 21, No. 1, pp 141 - 145.