A Framework for Software Engineering Education: A Group Projects Approach

Zaigham Mahmood

***Abstract*—** Software Engineering (SE) programmes at institutions of higher education aim to produce software engineering specialists who have the required knowledge of the SE tools, techniques and methods as well as the technical expertise to design and develop complex software. These programmes are generally well designed, however, those completing such programmes do not necessarily possess the required skills because of several inherent issues. This paper presents a framework for the provision of SE education using a *Group Projects* approach and suggests that there is a need to provide opportunities for students to work individually and in pairs in their first year of the course, in groups of 4-6 in the second year and in larger groups of at least 10 in the final year. Discussing the issues, the paper presents solutions to some of the difficulties that are often encountered with respect to team working, in particular: team formation, allocation of projects, group dynamics and project management and assessment. The aim is to put forward proposals to improve the effectiveness of SE programmes.

***Keywords*—**Group Working, Team Work, Group Project, Software Engineering, Computing

I. Introduction

Software Engineering (SE) is the design and development of complex software using accepted engineering principles. This includes rigorous analysis of users' requirements, functional specifications and software testing to ensure that the final product conforms to the requirements specification and satisfies users' needs [1]. Thus, university courses in SE aim to provide opportunities for students to learn the required theory as well as acquire the necessary skills to design and develop high quality software.

SE programmes are usually well designed and delivered. However, there is often a lack of time on a typical undergraduate course for students to acquire the necessary skills to engineer large software systems [2]. In the present work, it is assumed that undergraduate courses are of three or four year's duration and there are two semesters in each year.

The programming element of the courses i.e. the teaching of software design, development and testing using appropriate tools, generally follows the scheme given below:

- The first year concentrates on teaching a basic design technique and introductory programming following a simple software development paradigm. Students write simple programs working mainly individually. Approach is often syntax oriented and there is not enough emphasis on quality or software development as an engineering activity [3].
- In the second year, students are given opportunities to work in groups to specify, design and develop larger programs but the teams are usually too small and the software projects are often unrealistic. This is normally the only major group activity where students bring together the knowledge gained thus far and develop a software system based on a given software development paradigm.
- The final year usually concentrates on teaching the theoretical and advanced aspects of SE in greater detail. Often there are no opportunities for students to work in larger groups to engineer large and realistic software systems. There is normally a substantial research based module but this is not necessarily on software development.

This lack of opportunities for programming-in-the-large and programming-in-the-many in the final year is sometimes due to the fact that many students may not have acquired the necessary knowledge or skills required for a larger group activity. It is also due to the difficulties inherent in the following:

- Finding appropriately large and realistic projects.
- Completion of projects within the given space of time within the programme.
- Management and assessment of such projects.

In discussing the above, this paper suggests that SE courses need to be designed and delivered in such a way that students have opportunities to do at least the following:

- Design and develop small programs working individually and in groups of two in the first year.
- Specify, design, develop and fully test small to medium sized software systems working in teams of 4-6 in the second year and.
- Specify, design, develop and fully test small to medium sized software systems working in teams of 4-6 in the second year and.

This study also suggests solutions to some of the difficulties that are often encountered and puts forward proposals with a

Manuscript received March 14, 2007; Revised received October 3, 2007.

Z. Mahmood is with the School of Computing, University of Derby, DE22 1GB, UK (phone: 00-44-1332-591733; e-mail: z.mahmood@derby.ac.uk).

view to improving the effectiveness of SE programmes.

II. Group Working In The First Year

Software design, development and testing is at the heart of SE. In the first year, students learn programming-in-the-small or what is traditionally known as 'programming'. Here, they learn a design method, a computer language and develop small programs. The teaching approach often depends too heavily on the chosen language where the emphasis is on the syntax of the language rather than the modeling of realistic computational problems [2]. The approach does not teach development of software, as an engineering activity, because of a distinct lack of emphasis on accepted engineering principles and, in most cases, there is seldom a programming activity where students work in teams. It is essential that students begin to understand, right at the start of the course, at least the following [1]:

- That SE requires the use of engineering principles for the construction of programs.
- That SE implies working in teams and therefore effective team working is important.
- That SE refers not only to writing statements in a computer language, but the entire process of specification, design, coding, testing, implementation and documentation.

It is suggested that, in the programming modules, student's programming skills should be assessed by means of programming projects following the scheme given below:

- Students work in groups of, ideally, two.
- They understand the given specifications and decide on solution strategies.
- They design programs using a simple design approach.
- They write programs using a modern language, following given coding standards, and exhibiting good programming practices.
- They write programs using a modern language, following given coding standards, and exhibiting good programming practices.

Formation of pairs can be student's choice. Program specifications should be provided by the tutors and properly constructed. Submission of students' work can be in two stages: 1) submission of design and 2) submission of the remaining project after receiving feedback on design. At this point, tutors can provide correct designs for subsequent program development. When assessing, emphasis needs to be placed on the quality of design, correct functionality, completeness of test plans and evidence of good programming practices.

III. Group Projects In The Second Year

At the second year level, a majority of the SE programmes include a module where students work in small groups to carry out an entire SE activity. The aim is to provide opportunities for students to work in teams. Students are required to conduct the entire software development cycle, manage their own time, learn to respect others' points of view and understand the social and ethical issues involved [1, 3]. Such a module is an essential component of an SE course, however, there are a number of inherent issues. We find that: 1) projects undertaken are usually too small and often unrealistic, 2) final products are not properly designed and constructed and 3) Project and time management are often unsatisfactory. There are several reasons for these problems, including the following:

- Students find it difficult to work in groups.
- There is sometimes a lack of commitment on the part of the students.
- Students lack the experience of time and project management.
- Tutors find it difficult to have enough realistic projects.
- Often there are personality clashes and other problems between team members.
- The assessment and monitoring of projects is far from easy.

It is essential that students work in groups to learn group dynamics, experience project and time management and enhance their inter-personal skills [3]. In real life, they will often work in large teams so the experience gained here will be extremely invaluable. Some institutions use team projects throughout the course to simulate a real working environment and provide students with transferable skills [3, 4]. Thus, for a *Team Project* module in the 2nd year, it is suggested that:

- Students work in groups of ideally 4-6 students
- Each group produces a requirements document, based on the proposal suggested by the group itself, and submits as the first task for assessment
- Each group produces a system design based on the requirements specification, using a design method they have already studied, and submits as the second task
- Each group develops a software system based on the design, following an agreed coding standard, using a modern language, ensuring that quality criteria have been met, and submits as the next task
- Each group produces test plans and fully tests the system to ensure that functionality conforms to the requirements specification and submits as the next task
- Each group demonstrates the software system and submits a detailed project report for the next task and the final assessment.

The following sections provide more detailed information with respect to the above.

A. Team Formation:

Formation of groups can be student's choice, however, it

would be useful if each group has at least one programmer, an information systems specialist and one student with some knowledge of human factors or user interfaces. Initially, groups should be invited to suggest projects and submit brief project specifications. The projects should be finalized after discussions with the tutors, ensuring that they are at the right level.

B. Project Specification:

First stage should be the production of a complete requirements specification following the guidelines provided by the tutors. Students need to be realistic when agreeing projects' features and facilities. Design phase should not be started until the specification has been submitted and discussed with project tutors. Test plans must also be produced as part of this stage [1].

C. Software Design:

Second stage would be to produce a complete design, using a design approach that students are familiar with. Students need to be aware that design is independent of the computer language. The next phase should not be started until the design has been submitted and discussed with project tutors.

D. Programming:

This is the third stage to implement the design using a computer language. Students need to be reminded that the quality of code, use of appropriate constructs, implementation of engineering principles and other aspects such as readability, re-use and maintainability must be evident in the final product.

E. Verification and Validation:

The next stage is to conduct verification and validation of the software system to ensure that the system conforms to the requirements specification submitted earlier and that the software produces correct results.

F. Product Demonstration:

The fifth and final stage is the demonstration of the system, submission of the final product and a detailed project report for further assessment. The presentation will provide opportunities for students to show the fruits of their efforts and provide detailed explanation of different aspects of the project activity.

G. Project Management:

Team Project activity should be regarded as independent study where students assume full responsibility. However, a timetable slot should be allocated so that there is a place and time where students can meet. Project tutors can also use this time to discuss students' progress and resolve difficulties that students may be facing. The final demonstration is the vehicle for project tutors to know exactly what contribution each member of the groups has made and thus help them when allocating individual grades to students.

H. Project Submission:

Several submissions at different stages of the project are extremely useful. This helps students to be organised and provides opportunities for them to conduct the SE activity in an appropriate manner.

I. Individual Assessment:

Assessment can be a difficult process. Often it is easy to grade the entire project but there needs to be a way of allocating grades to individual members of each team. Peer assessment is a useful device. At various points throughout the projects activity, groups should be encouraged to comment on other groups' deliverables. At the end of the project period, students should be asked to submit not only the minutes of the meetings of the groups where each member's activity will be recorded but also to allocate a percentage of work to each member. The final presentation is useful as the tutors can ask questions to ensure which member of the group has done exactly what.

IV. Software Projects In The Final Year

Most SE programmes do not provide opportunities for students to design and build large software systems, working in groups. The emphasis appears to be on providing in-depth knowledge of various theoretical aspects of SE. However, in order to understand and appreciate the inherent complexities and issues associated with SE, it is essential that students engage in the development of medium-to-large software projects working in teams of many, using as much as half of their time in the final year for this activity [2]. Some of the reasons for not having such a mechanism in the final year are given below:

- Often, there is not enough time to conduct a large project. It is also difficult to find reasonably large realistic projects.
- Management of large teams is problematic.
- Students may not have appropriate knowledge and skill at the start of such projects.
- Assessment of projects and allocation of individual grades is not easy.

Not withstanding the above, it is important that an SE course has a *Software Project* module in the final year where students work on much larger projects in teams of at least 10 students [2]. This suggests that this module is several times the size of a standard module and runs in the second half of the final year. Here, larger groups of students should be required to work with minimum supervision and on much bigger software projects using languages, tools and methodologies of their choice. Hopefully students will have studied these in the previous years of their programmes.

It is suggested that students should be organized into groups and allowed to choose their projects in semester 1 (the first half of the final year). They should also do some preliminary

work but do not start their projects until the start of semester 2 (second half of the year). Then, they devote the entire semester 2 carrying out the project activity. Each project team should start with a given idea of a specification, proceed through the stages of requirements specification, project planning, design and implementation and then produce a reliable and stable software system, which they submit for assessment. Producing a project report and demonstration of the final system are essential requirements of this module. In this module, the emphasis is on:

- The quality of deliverables
- The ability to satisfactorily proceed through all stages of a software development process
- The ability to apply the required knowledge and transferable skills
- The ability to use the techniques and tools as appropriate
- Effective group working, inter-personal skills and professionalism.

It is essential that students use a formal notation for software specification and appropriate design and implementation tools.

It is important that teams are reasonably large and software projects are at least medium sized. Unless the students are exposed to projects of reasonable magnitude, they will not be able to appreciate the inherent complexities and issues relating to programming-in-the-large and programming-in-the-many [1]. However, project size should be such that students can easily complete them within the limitations of the programme structure. Projects should be real, though not necessarily for real clients: working for real clients may have legal implications if money is involved and projects are not satisfactorily concluded. If the course team has good industrial links then finding good projects may not be a problem. Groups should be chosen by the tutors but students should have the freedom to choose any projects they like. Once groups and projects are allocated, they should not be changed unless an exception situation requires a different course of action.

Each project should have a Project Management Team (PMT) consisting of one main supervisor and two secondary supervisors. Students should work with the minimum of supervision. However, for advice and help, they should not be restricted to the PMT: they should be allowed, in fact encouraged, to see the members of staff who are most experienced to provide the required guidance. In this way, the entire teaching team will be directly involved with all projects and all students. In all other respects, the scheme described for the group project module in the second year, as mentioned above, can be applied - including several submissions at different points during the project activity and the milestones.

Assessment of projects and allocation of marks to individual students is far from easy. It is the responsibility of the PMT, and thus the entire course team, to ensure that assessment is appropriate and the marks given to individual students are fair. For each project, marks should be given by their PMT on the basis of the final product, project report, demonstration, presentation, individual effort and contribution, students' time management and their commitment and attitude. Marking scheme can be similar to the one used for the group project in the second year. It is essential that the course team has a well thought out procedure for monitoring, management and assessment of these projects. Usually, the final classification depends heavily on the result of final year projects.

There is no doubt that working in large groups is difficult for students. It is equally difficult for PMT to manage and assess large group activities but, as mentioned before, unless students are exposed to working in large teams and developing large software, they will not be able to understand and appreciate the inherent complexities associated with SE.

V. Conclusion

University programmes in SE aim to produce well-qualified software engineers. Whereas, students completing such programmes have the required knowledge of the engineering principles and the methods, techniques and tools, they do not necessarily possess the relevant skills of specification, planning, design and implementation. This paper mentions the reasons for this and discusses, in some detail, the importance and provision of group working component at all levels of such programmes. Discussing the issues concerning programming-in-the-large, this paper presents a framework for the SE education using a *Group Projects* approach and suggests that SE courses should provide opportunities for students to work individually and in pairs in their first year of the course, in groups of 4-6 in the second year and in larger groups of at least 10 in the final year. The paper also suggests solutions to some of the difficulties that are often encountered with respect to team working, in particular: formation of student groups, allocation of SE projects, internal dynamics of group working, management of projects and assessment of student's performance. The aim is to put forward proposals to improve the effectiveness of SE programmes.

References

[1] Sommerville I. (2006), "Software Engineering", 8th Edition, Addison Wesley, UK

[2] Mahmood Z. (1995), "Core Requirements for a Degree Course in Software Engineering", *Proc Int Conf on Software Engineering in Higher Education (SEHE95), Alicante, Spain;* Computational Mechanics Publications, 1995.

[3] Marchewka J T. (2005), "Information Technology Project Management", Wiley Int. Edition, UK

[4] Conway D E, Dunn S C. "BCS and IEE Accreditation of Software Engineering Courses", *Software Engineering Journal*, 4(4), 1989.

Dr Zaigham Mahmood is a Senior Lecturer in the School of Computing, University of Derby, UK. He has an MSc in Mathematics, an MSc in Computer Science and a PhD in Modeling of Phase Equilibria. He is also a Chartered Engineer and a Chartered Information Technology Professional.

Dr Mahmood has many publications. His research interests are in the areas of software engineering, project management, software metrics and process improvement.