Team Projects as a Means to Learn Some Aspects of Development Process of Information Systems

Mária Bieliková*
bielik@elf.stuba.sk
Pavol Návrat*
navrat@elf.stuba.sk

Abstract: Team projects play an important role in education of software engineers. A typical project aims at developing an information system for a specific environment or organisation. While working on the project, students apply information system modelling techniques and use appropriate tools, such as CASE tools for analysis and design of the information system. We report on such a team project module that is part of a postgraduate course in Informatics.

Key Words: Team, education, information system.

1 Introduction

Analysis and design of information systems is an important part of many software engineering curricula. Frequently, a subject devoted to this topic is included in a curriculum. Essential is mastering an appropriate methodology supported by the use of corresponding CASE tools. Bajerski [1] reports in detail about experience with using such tools.

Another important aspect is the way in which formal methods are integrated into the curriculum. Formal models of software systems and logics are used to characterise and reason about their properties. At CMU [6], the masters course in software engineering includes topics such as state machines, algebraic models, process algebras, trace models, compositional mechanisms, abstraction relations and temporal logic.

However, learning all this does not suffice for the graduates to be appropriately prepared to face the challenges of today's information systems industry. There is a growing recognition that software development based on teams is critical in managing the complexity of developing software systems. The question is how this experience reflects in software engineering curricula. In the following overview, we try to summarise some of the efforts to integrate team-based development in software engineering courses.

One of perhaps the most distinguished features of the CMU course is the software development studio, which runs throughout the whole four-semester course. Students work closely with instructors and other students to carry out a significant development project. The concept of a software engineering studio has been introduced into their curriculum also by others [17].

Information systems development is taught in a form of a practicum as reported by [7]. Students are divided into groups of 6 to 10. The practicum comprises three phases: contract, project, and evaluation.

* Slovak University of Technology, Department of Computer Science and Engineering, Ilkovičova 3, 812 19 Bratislava, Slovakia

In [2], an approach to building and managing a team learning environment is described. Teamwork is applied throughout the whole course. The course is designed as a software engineering curriculum framework including not only subjects but also tools to support teamwork such as collaboration tools, team documents, help desks, libraries of resources, and a recognition system for identifying high performance in teamwork.

Another possibility of incorporating team work into software engineering curriculum relates to a novel module entitled "Setting up and Running your own I.T. Company" at the University of Sheffield [9].

While many aspects of these approaches are attractive, it may be difficult to adopt them because each module is designed within a specific context. We discuss variability of different methods to implement such a module.

2 Team Work Dynamics

An objective of any good curriculum design is to prepare the graduates for their envisaged professional careers by providing them an appropriate education. In software engineering, the essential feature is that the graduates will develop and maintain software or information systems of such a complexity that these tasks cannot be handled at an individual level. Such tasks can only be accomplished in groups or teams. The engineers need to join their efforts to be able to complete the tasks in a realistic time. Moreover, they usually need to join with experts in other fields to be able to understand and analyse the problem domain and to design a system solution for it.

One comment is appropriate to explain the difference that is sometimes made between a group and a team. Both groups and teams comprise individuals who have a common task and who collaborate to accomplish it. A team is distinguished by complementary efforts of its individuals that induce a synergic effect.

Clearly, it is more difficult to facilitate formation and proper functioning of teams of students, but it is also more desirable because this form of collaboration is in all aspects superior. Therefore, our goal has been to approach as much as possible in devising our team project module to the above interpretation of the concept.

In designing the team project as a part of a curriculum, several aspects should be considered. We shall focus on the following aspects: team formation, team communication, team assessment, problem statement and assignment, development process and organisational aspects. Each aspect is discussed separately. First, several different alternatives are outlined, their pros and cons are analysed and then our approach is described.

2.1 Team Formation

The way teams are formed can have a great influence on a project. There are two basic methods for forming teams: students create teams by themselves or students are assigned to the teams. Several variations of these methods can be adopted for team projects. For example, assigning students to teams can follow different goals: a) to equalise the differing teams' project specific knowledge and skills, b) to balance team membership according to differing psychological profiles (using the results from known psychological studies) [18]. Hilborn [8] reports on experience with following both goals. We could find also one module with a random assignment of students to teams [14].

If the students can sign up for teams, there is a tendency to base the choice on a personal acquaintance. Team efficiency can be in such cases high in some teams. On the other hand, there is a risk that team quality can vary quite significantly across teams. This can be improved by forming teams in two steps. First, students assemble teams. Then, supervisors rearrange those teams where an undesirable mixture of skills can be recognised [3]. A justification for this procedure is that to work with people one already knows and is a friend with is only one of possible scenarios encountered in practice. It may not be a prevailing characteristic of teams formed ad hoc in industrial conditions for a particular project [2].

Our approach is to let students assemble their teams. The advantage is its simplicity. We are prepared to step in and make rearrangements when big differences between teams become apparent. Acquaintance based teams often overcome, or at least tend to minimise a number of problems which can arise in a team life cycle: little effort is needed to get teams started; scheduling and arrangements of out-of-class meetings is easier as the students know each other well and often have similar preferences. Moreover, in many cases, there is more pressure on each member of a team “not to let down one’s friends”.

After two years of running the team project module we have observed that the distribution of excellent students was more or less even.

One particular aspect concerns the size of a team. The bigger is a team, the more difficult it is for its members to participate fully in its activities. In a big team, there are formed subgroups that start to work independently. Individual roles in a team become fuzzy and they overlap which is a source of possible conflicts. It is more difficult for a team leader to perceive individuals and atmosphere in the team.

Our experience endorses the frequent suggestion to have the size of a team between 4 and 8. If the size is smaller, team’s creativity and flexibility decreases. If the size is bigger, number of interactions among its members rapidly increases and the team becomes less manageable.

2.2 Team Communication

Students must decide how to mesh their various work patterns to develop the required product. Basic skills which should be exercised and monitored include time planning, product presentation, decision making, keeping personal log on work hours, negotiating and keeping work contracts, managing priorities and deadlines, acquiring information (e.g., by interviewing people) and accessing information (e.g., by searching documents).

To facilitate communications among team members (and also among teams) there is more and more often used internet or intranet. We encourage this practice, not to speak about such elementary means of electronic communication as e-mail, which is indispensable. Moreover, each team creates and regularly maintains its web page that includes also their “official” presentation of their project.

Students within a team should meet on a regular basis. Teams meet with their project supervisor at least once a week. There are also regular private meetings of the team. All major decisions should be recorded in a project log.

We adopted an approach where students should experience various responsibilities for different parts of the system that is being developed. Distributing the work evenly among the team members is a precaution against a possible excessive individual workload and also against problems in case when a team member quits for whatever reasons.

2.3 Problem Statement and Assigning Problems to Teams

Problem complexity, its breadth and depth are influenced above all by the expected workload attributed to the module. Problems to be solved should meet the criterion that they are too complex for one or even two excellent students to do all the work needed to complete team’s assignment.

Information systems development is a frequently used and recommended theme. In our team project module this year, we offer three kinds of information systems development projects, one kind of project of a simulation software system development and one kind of project that involves development of an educational program in physics.

A team's assignment can be such that each team works on different projects during each standard project completion phase. For example, a team designs a system that was defined in another team's specifications. It implements another team's design, and tests still another team's implementation. This approach resembles "real world" development and causes students to participate seriously in all phases. Students realise the importance of a well developed documentation, possibility of incorporating changes, etc. Adopting such approach is recommended only if the module is spread to at least two semesters. One difficulty with such a pipelined approach is that it is very demanding. Students must swiftly become knowledgeable of several different domains.

One problem can be assigned to more than one team. The advantage is increase of competition, more competent mutual reviewing, more intensive and focused class discussions among several teams, more transparent evaluation and assessment. We have positive experience with such approach.

Similar methods as for assembling teams can be adopted for assigning problems to them. The basic possibilities are: a) assign randomly; b) get teams to bid for projects; c) let teams indicate their preferred choices; e) use a first come first served policy.

We use the second method. At least some of the teams will work on the problem in which they are interested and often have previous experience. This is probably the most realistic approach. Things become a little bit complicated be the fact that the bidding period causes a delay in the actual start of the project work.

2.4 Team Assessment

Methods of team assessment should be based on some objective criteria. Based on them, a supervisor is usually responsible for determining students' grades. Because of the difficulties involved in evaluating individual student performances, often student peer evaluations are used (as a help) [4, 19].

Our assessment of teams comprises the following components: product quality, activity, documentation, reviewing, presentation and answers to questions on product and software engineering methods used.

A supervisor assigns grade for the team as a whole and then divides points to individual team members. Individual assessment is based on the student peer evaluation. We use several methods for students peer evaluation: a) students are asked to order their team mates; b) students assess each of their team mates on a predefined (e.g., 10) point scale; c) students determine part of work performed by each team mate. The last method seems to be most effective.

2.5 Development Process

Composition and shape of the development process for a particular project is mainly influenced by the problem domain and project resources. In spite of this, phases of a classical software life cycle can be considered: project definition, analysis and specification, design, implementation, testing and maintenance. Usually life cycle ends with implementation and module testing phase. But there are reported approaches where a maintenance phase is involved. For example, a module can consist of two parts: the first part is the development of a new system, and in the second part the teams are asked to make certain modifications to those systems [15].

Our experience is that the most important factor is the time (how long the project lasts and how many hours per week it is allocated). If a team project module is just one out of several modules taken concurrently, covering all phases of the software life cycle in 2 or 3 semesters is hardly possible. From the point of design of the module, stress should be put on the team process, communication, etc. rather than on concentrating on the particular phases of software development. We warn against expecting too much from the team project module in the sense of teaching students new methods, tools or technologies.

Table 1 lists main activities performed in a two semester Team Project module at the Slovak University of Technology as adapted for a project of developing an information system.

Table 1. Team Project Activities and Schedule.

Week	Activity
1	Teams formation, information about various problems statements offered.
2	Elaboration of a bid for a problem selected by teams, requirements acquisition from a client.
3	Team presentation of a bid, evaluation of bids and assignment of problems to the teams.
4	Problem analysis.
5,6	Requirements specification and architectural design.
7	Delivery of requirements specification document. 1st check point of the WWW presentation of the project.
8	Teams peer review of requirements specification.
9	Amending of the shortcomings found.
10	Design of a prototype (selected parts of a system, e.g. user interface for information systems projects).
11	Implementation of a prototype.
12	Delivery of a prototype together with documentation. User oriented (black box) presentation, questions and answers.
13	Team peer review of a prototype.
14	Evaluation of the results achieved in autumn semester together with amending of the shortcomings found. 2nd check point of the WWW presentation of the project.
15, 16	Detailed design, plan of an integration, design of test data.
17, 18	Implementation and testing of software modules.
19-21	Implementation and testing of software modules, gradual integration.
22, 23	Product integration and testing.
24	Completion of the documentation. Last minute tests, software packing for delivery.
25	Delivery of the software system (programs and documentation). Team peer review of the delivered system. 3rd check point of the WWW presentation of the project.
26	Technical (white box) presentation, questions and answers. Active participation in peer team presentation, questions and answers.

Each project must be presented in a seminar, to an audience of interested staff and students (mainly those who worked on similar assignments). An important part of the development process is reviewing the work of another team. For each team a competing team is assigned. Students should write a review document with its structure specified beforehand. Results are then discussed in a joint meeting where students of one team are supposed to present their review and respond to another team's review of their work. Different kinds of review can be adopted with different frequency. The main constraint here is the length of the module. We adopted approach where projects are reviewed after completing each development phase: specification review, review of a prototype, design review and review of an implemented product (including review of a source code).

2.6 Organisational Aspects

Organisation of the team project is influenced mainly by class size of the module (very focused module for 10-25 chosen students [5] versus large-scale module of more than 300 students in 50 teams [3]). In fact, the size of the class determines choice and organisation of the supervisors. For example at the Vienna University of Technology 5 professors, together with 20 senior student supervisors, guide 50 teams of 5-7 students. A professor takes the role of a middle management, a supervisor acts as a quality assurance person, and team members are to work as project management and software engineers.

It is helpful when the module is accompanied by lectures on project management, team work, quality assurance, etc.

Figure 1 shows roles and numbers of people involved in our module during the current year (1998/99). For issues that concern the whole module, all supervisors meet with a head module supervisor about once a month or as the situation demands.

3 Discussion and Conclusions

From the above it follows that the idea of incorporating a team work practice usually in a form of some kind of a team project has been adopted by and large. However, there are many issues that remain open. To name a few, organising projects as real-life ones mirroring the industrial conditions while solving problems defined by an industrial partner is discussed as clearly a desirable, but hardly a completely realistic goal. On the other hand, curricula designers should be careful when relying solely on academic environment and should seek at least some kind of an industrial involvement. Valuable experience from running industrial projects is described in [20].

Assessing project work of each individual member of a team is not a simple matter as it is sometimes not possible to identify a personal performance. Rather, marks can be seen as a quality evaluation of the team's product [17] so all team members get the same mark.

Another question is whether or not students should be assigned to a given team. Positive answer is supported by the fact that a similar method is typically applied in a real-life setting.

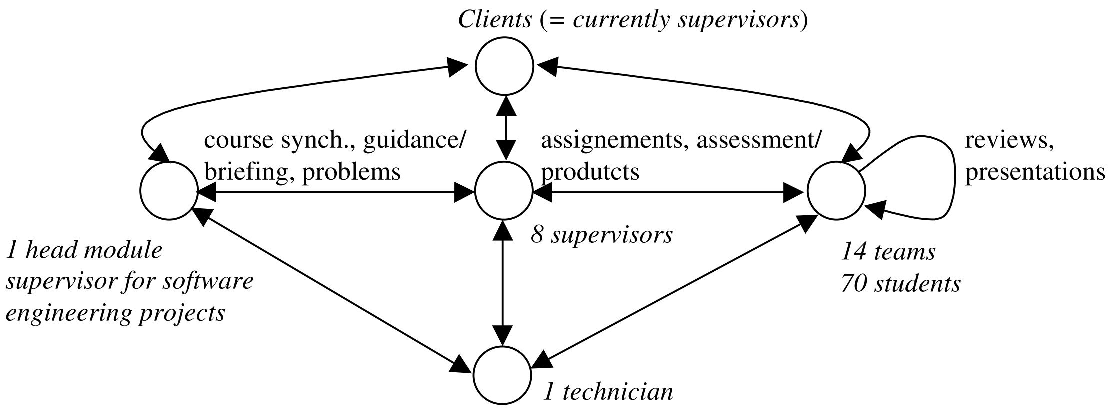

Figure 1. Roles and numbers of people involved in Team Project.

Number of students in a team varies slightly at different institutions, but never has been reported to be less that 4.

A team project is increasingly being considered essential in educating software engineers. At the postgraduate level, it is practically indispensable. At the undergraduate level, it has been introduced recently into curricula at various institutions, e.g. Poznan University of Technology. As Lukasik [10] reports, it is a one-semester project for teams comprising 4-5 students. However, from the masters degree student's perspective, it is a three-semester endeavour. According to Nawrocki [13], in Poznan, they have adopted the approach of assigning the postgraduate students the role of project managers. In the first semester, they, working in pairs, elaborate requirements and rough design, estimate effort, plan work and form a team from undergraduate students. In the second semester, they work together in teams of 6 on detailed design, implementation, testing and integration. During the subsequent industrial internship, the postgraduate students perform on-field testing of the system in an industry setting, train prospective users and fix bugs. In the third semester, they are responsible for software maintenance. The underlying idea is that these engineers should be prepared to develop and apply latest results of science and technology throughout their careers. The education should help them become managers and leaders of software teams.

A similar approach is followed in Prague [16] where also teams are formed by combining undergraduate and postgraduate students with the latter being assigned some of the managerial tasks. The project takes 2 semesters.

In design of our curriculum [11, 12] we followed an approach where students first learn the required knowledge and skills individually or in small groups (2 or 3 students), and then make use of it in a team [5, 3].

The team project module runs for the second time with 17 teams in the first year and 14 teams this year. Much stress was put on the supervisors who should manage the teams and help them change from a supervisor-managed mode to a self-managed mode. Quite often students voiced the desire having a bigger diversity of themes, including problems which would be defined by themselves. Some students questioned the value of work that went beyond programming, especially of writing documentation. This is a broader problem with underestimating the documentation in the projects. In line with others' experience [3], most teams complained that the module demanded significantly more work than they were credited for in their curriculum. In spite of initial problems the students responded well to the challenge and several commented that this was one of the most exciting activities of their student career. Most of them valued experience with a system integration.

This work has been supported by the Slovak Science Grant Agency, project No. G1/4289/97 "Methods and tools of software systems development".

References

1. P. Bajerski, K. Stapor and M. Wycislik: *Experiences of using CASE tools in teaching of analysis and design of information systems.* In Proc. of SEES'98 - The Software Engineering Education Symposium, P. Klint and J.R. Nawrocki (Eds.), Scientific Publishers OWN, 1998, pp.49-53.
2. S. Becker and A. La Salle: *Translating software engineering principles into practice: team frameworks.* In Proc. of SEES'98 - The Software Engineering Education Symposium, P. Klint and J.R. Nawrocki (Eds.), Scientific Publishers OWN, 1998, pp.54-61.
3. S. Biffl and G. Thomas: *Preparing students for industrial teamwork: a seasoned software engineering curriculum.* IEE Proceedings, Vol. 145, No. 1, February 1998, 1-11.

4. J. Brown and G. Dobbie: *Software engineers aren't born in teams: supporting team processes in Software Engineering project courses.* In Proc. of Int. Conf. Software Engineering: Education & Practice, M. Purvis, S. Cranefield, S. MacDonell (Eds.), IEEE Computer Society, 1998, pp.42-49.
5. R. Dawson and R. Newsham: *Introducing software engineers to the real world.* IEEE Software, Vol. 14, No. 6, 1997, 37-43.
6. D. Garlan, D.P. Gluch and J.P. Tomayko. *Agents of change: educating software engineering leaders.* IEEE Computer, Vol. 30, No. 11, November 1997, 59-65.
7. J. Gordijn and F. Niessink: *Balancing freedom and supervision in a real-life educational project.* In Proc. of SEES'98 - The Software Engineering Education Symposium, P. Klint and J.R. Nawrocki (Eds.), Scientific Publishers OWN, 1998, pp.77-84.
8. R.B. "Sam" Hilborn: *Team learning for engineering students.* IEEE Transactions on Education, Vol. 37, No. 2, May 1994, 207-211.
9. M. Holcombe and A. Stratton: *VICI: Experiences in introducing student run software companies into curriculum.* In Proc. of the Project 98 Workshop, Projects in the Computing Curriculum, M. Holcombe *et al* (Eds.), Springer, London, 1998, pp. 103-116.
10. E. Lukasik: *Team project – a midpoint in teaching software engineering.* In Proc. of SEES'98 - The Software Engineering Education Symposium, P. Klint and J.R. Nawrocki (Eds.), Scientific Publishers OWN, 1998, pp.248-252.
11. P. Návrat and M. Bieliková: *The context and the contents of software engineering education in Slovakia*. In Proc. of SEES'98 - The Software Engineering Education Symposium, P. Klint and J.R. Nawrocki (Eds.), Scientific Publishers OWN, 1998, pp.141-148.
12. P. Návrat and Ľ. Molnár: *Curricula transformation in the countries in transition: an experience from Slovakia.* IEEE Transactions on Education, Vol. 41, No. 2, May 1998, 88-91.
13. R.J. Nawrocki: *Towards educating leaders of software teams: a new software engineering programme at PUT.* In Proc. of SEES'98 - The Software Engineering Education Symposium, P. Klint and J.R. Nawrocki (Eds.), Scientific Publishers OWN, 1998, pp.149-157.
14. L.R. Neal: *Using group project work to enhance the learning of professional issues.* In Proc. of the Project 98 Workshop, Projects in the Computing Curriculum, M. Holcombe *et al* (Eds.), Springer, London, 1998, pp. 31-43.
15. L. Ohlsson, C. Johansson: *A practice driven approach to software engineering education.* IEEE Transactions on Education, Vol. 38, No. 3, August 1995, 291-295.
16. K. Richta, P. Slavik and Z. Mikovec: *Development of software engineering educational model at CTU Prague.* In Proc. of SEES'98 - The Software Engineering Education Symposium, P. Klint and J.R. Nawrocki (Eds.), Scientific Publishers OWN, 1998, pp.164-169.
17. P.N. Robbilard: *Teaching software engineering through a project-oriented course.* In Proc. of the 9th Conf. on Software Engineering Education, April 1996, 85-94.
18. J, C. Schlimmer, J.B. Fletcher and L.A. Hermens. *Team-oriented software practicum.* IEEE Transactions on Education, Vol. 37, No. 2, May 1994, 212-220.
19. T.J. Scott, L.H. Tichenor, R.B. Bisland and J.H. Cross: *Team dynamics in student programming projects.* SIGCSE, No. 4, 1994, 111-115.
20. A. Stratton, M. Holcombe and P. Croll: *Improving the quality of Software Engineering courses through university based industrial projects.* In Proc. of the Project 98 Workshop, Projects in the Computing Curriculum, M. Holcombe *et al* (Eds.), Springer, London, 1998, pp. 47-69.