Enhancing Software Projects Course Work by Advanced Management

Jyhjong Lin
Department of Information Management
Ming Chuan University
Kweishan, Taoyuan County, Taiwan 333
E-mail: jlin@mcu.edu.tw, Fax: 886-3-3593875

Abstract: - Software projects are an important course in software engineering curricula. They provide students with an opportunity to gain valuable experiences in applying the theoretical materials learned at software engineering courses. However, many drawbacks in current projects course work make these benefits difficult to realize. In this paper, we discuss these drawbacks and how they affect the effectiveness of this course. To address these difficulties, we advocate a model of enhancing the management approach to provide an effective administration for each project. Our experiments from the model have shown significant improvements in the quality of projects and the experiencing effects of students.

Key-Words: - software engineering education, software project, management, model

1 Introduction

Software projects are an important course in software engineering curricula. They provide students with an opportunity to gain valuable experiences in applying the theoretical materials learned at software engineering courses. Similar to many other institutes [1-5], the Department of Information Management at Ming Chuan University [6] has offered a student software projects course for this purpose for many years. This course contains a software project for each five-member group of students with a two-semester period from the second half of their third study year to the fist half of their fourth study year. The project requires the team to develop/maintain and document a software system under the supervision of a departmental faculty who provides mainly professional and technical assistances to the group. The supervisor determines the topic and context of the project and keeps track of the project progress by regular meetings with the group, reviewing written documents, and solving technical problems.

After performing this course for several years, we find however that there exist many drawbacks in the current work which make students difficult to benefit as largely as we expect. According to our observation and analysis with students, these drawbacks include:

1. Since there are a number of projects for covering such many students in the department (two hundred and forty students to be supervised in each year), every departmental faculty is required to supervise a couple of project teams. The problem behind this assignment is that many of departmental faculty are not software engineering experts and hence unable to assist their project teams effectively on software engineering issues; team members cannot benefit largely from them to enhance their software engineering skills.

2. Each project is undertaken under an individual supervision; its development process and phased deliverables are somewhat followed and developed in a different manner from those of other projects. Such different manners always confuse students of different project teams and further seriously twist their knowledge on the real values of this course.

3. For some reasons, current projects are almost the development of software systems from scratch. We find that very few of them address

the maintenance or re-engineering of software systems which is however important also in software engineering issues. This ignorance makes students have weak experiences in these subjects.

4. In developing software systems from scratch, since these projects may refer to unknown application domains or require new implementation tools, students produce usually systems that are not complete or error-free to achieve commercial realities. Continuous enhancements on these systems in subsequent years to achieve better commercial realities are somewhat good ideas but frequently unplanned or unnoticed by faculty individuals. Commonly these systems are at last threw away or left only as demonstrative pieces. This is badly impressive for students to recognize that such a projects course seems only to give some extended class assignments; it does not result in the delivery of truly useful software systems. This seriously impacts their appreciation on software engineering issues; they do not feel that applying software engineering materials is as essential or critical as claimed for developing useful software systems – very few useful systems are ever delivered from this course. In recent years, we find that students gradually resist on or are reluctant to apply the software engineering materials, and hence their experiencing effects gradually become deteriorated year by year.

To address these problems, we advocated two years ago a model of enhancing the management approach to provide an effective administration for each project. The model has three goals: (1) enhance the software engineering skills of students and their experiencing effects; (2) provide faculty members with effective project administration to improve the quality of projects; (3) engage in continuous enhancements on software systems to achieve better commercial realities.

After working with this model by involving experimental projects over two years, our experiences have shown significant improvements on the course work and achieving its goals. In the following sections, we overview first its related work in Section 2 and present then its details in Section 3. The experiences of working with it will be discussed in Section 4. Section 5 has conclusions and explains our future work.

2 Related work

Software projects are an important course in software engineering curricula. They provide students with an opportunity to gain valuable experiences in applying the theoretical materials learned at software engineering courses. In general, this course contains a software project for each group of students over a one- or two-semester period. The project requires the team to develop/ maintain and document a software system under the supervision of some supervisor(s) who provides professional or technical assistances to the group. For commonly recognized criticality of the software engineering paradigm, many campuses offer this kind of course with their respective emphases on its contents and lectures [1-5].

For instance, some practices for educational software engineering projects are introduced at University of Groningen in the Netherlands and Växjö University in Sweden [4]. Among these practices, in particular, each team is supervised and evaluated by two HoDs (Heads of Department by PhD students or teaching assistants) and a project supervisor. The HoDs are available for the students on a daily basis, while the supervisor meets students on a weekly basis. In this model, however, many problems arise that include e.g. complexity, focus on technology, and free riders. To address these problems, seven principles for good practices are presented and demonstrated to have positive efforts on educational software engineering projects. Another work can be found at Dresden University of Technology [2] where project teams are guided by senior students who in turn are supervised by a project course leader. Similar to the roles played by HoDs, these senior students work as tutors and are both consultants for the team members and customers for the software project. This work has therefore shown its effectiveness on software engineering project courses as in the above model.

After carefully analysis, La Trobe University set out a SE project strategy in 1993 for its software engineering projects course [1]. In this strategy, a teaching team acts as the top hierarchy of project management with a lecturer and up to four part-time associate lecturers to lead a software engineering project. All lecturers are trained to play several roles during the course, e.g. a team supervisor or team manager to assist in the allocation and scheduling of tasks, a tutor during the labs and consultations, a client to assist clarification of requirements, and a quality assessor during assessment time. In general, these lecturers are responsible for clear delivery of concepts and methodologies about software engineering theories and project management to the students. Based on a five-year experience, this strategy has shown its compulsory performance for the students to withhold professional capabilities of software engineering theories and practices.

In summary, many software engineering projects courses have been offered at various colleges at which students can gain valuable experiences about the application of software engineering theories on practical projects. As stated above, these existing courses organize students into groups to each complete a project under the supervision of project leader(s) which are usually involved directly by departmental faculty members or indirectly by designated senior students who in turn are guided by faculty members. Although the effectiveness of these ways have been demonstrated by several years experiences of practical projects, there are still some limitations that degrade their performance on the software engineering projects course:

(1) Assigning each faculty member to supervise some projects assumes itself that the faculty member has sufficient software engineering knowledge to assist the project team on various encountered software engineering issues. However, the assumption is not realistic since for any college department that each faculty member is a software engineering expert should not be expected.

(2) Almost all existing projects focus on the development of software systems from scratch. However, since such one- or two-semester projects usually refer to unknown application domains or require new implementation tools, the systems produced are not uncommon to be incomplete or error-prone and hence their maintenance or re-engineering for better commercial realities are often needed. These needs are nevertheless neglected by existing situations.

(3) Almost all existing projects are undertaken under individual supervisions (directly by faculty members or indirectly by designated senior students who in turn are guided by faculty members). Their development process and phased deliverables are somewhat followed and developed in a different manner from those of other projects. Such different manners may confuse students of different project teams and further seriously twist their knowledge on the real values of this course. However, very few discussions about this problem can be found where a departmental mechanism for defining such project materials as process, deliverables, and metrics is needed.

To address these limitations, we use a new management model that focuses on (1) enhancing the software engineering skills of students and their experiencing effects; (2) providing faculty members with effective project administration to improve the quality of projects; (3) engaging in continuous enhancements on software systems to achieve better commercial realities.

3 The course management model

This section describes the structure and procedure of our course management model. Figure 1 shows the structural aspect of the model. We explain it more detail in the next subsection. The procedural aspect will be discussed in subsection 3.2.

3.1 The Structural Aspect

In the structural aspect, as shown in Figure 1, some special interests groups (SIGs) are organized where each departmental faculty is required to join at least one SIG. Currently we have five SIGs from the fields on which our

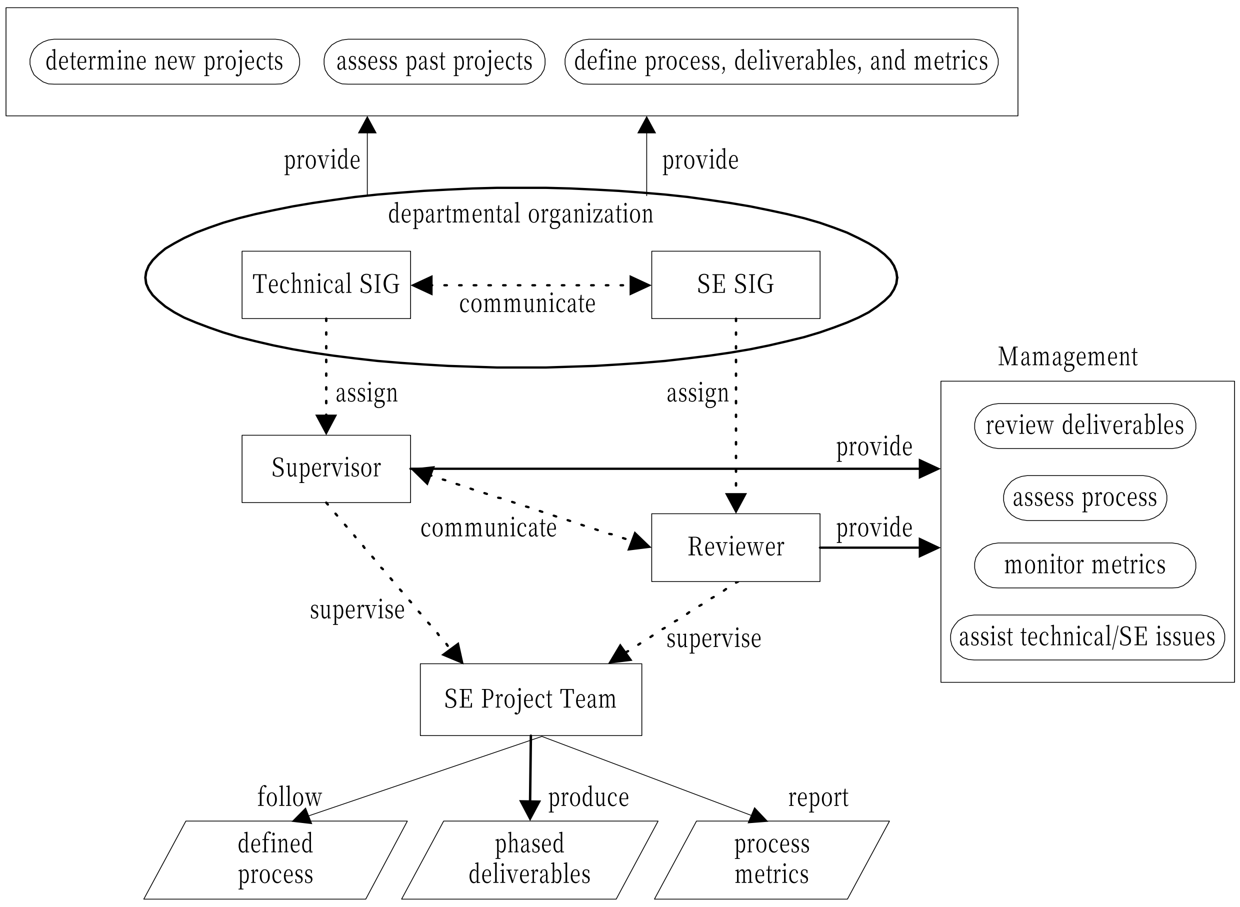

Figure 1: The Software Projects Course Work Model

department focuses: Electronic Commerce (EC), Business Intelligence (BI), Knowledge Management (KM), e-Learning (EL), and Web Application Software Engineering (WASE). In designing the projects course during the first half of each study year, these SIGs meet together to work out the following tasks: (1) assess past projects and their deliverables to determine whether any of them need to be enhanced or re-engineered for achieving better commercial realities; (2) based on either new ideas or requirements of enhancing/re-engineering past projects among our focused fields (i.e., EC, BI, KM, and EL), determine the topics and contexts of the forthcoming projects; (3) for each project, based on its subject and context, the WASE SIG defines a process and metrics to be followed and collected during the project period; (4) for each project, based on its field, assign a field SIG member to supervise it; and (5) for each project, assign a WASE SIG member to co-supervise it. Thus, in our model, all project matters to be followed/produced have been predefined by SIGs before student teams realize them. Both of the field and WASE SIG members cooperate to maintain the quality of the project as well as provide students with the assistance on field and software engineering issues to enhance their respective skills.

After each project has been allocated to a self-selected team, the team proceeds to accomplish the project through two semesters. During its period, the project is proceeded and supervised by the following events: (1) the project team follows a process defined by the WASE SIG; (2) the project team produces phased documents for being reviewed by its twin supervisors; (3) the project team reports those metrics defined by the WASE SIG for being assessed by its twin supervisors; (4) the project team gets needed field/WASE assistances from its twin supervisors.

3.2 The Procedural Aspect

Under the structure of the model, the projects course is performed as shown in Figure 2 with the following procedure:

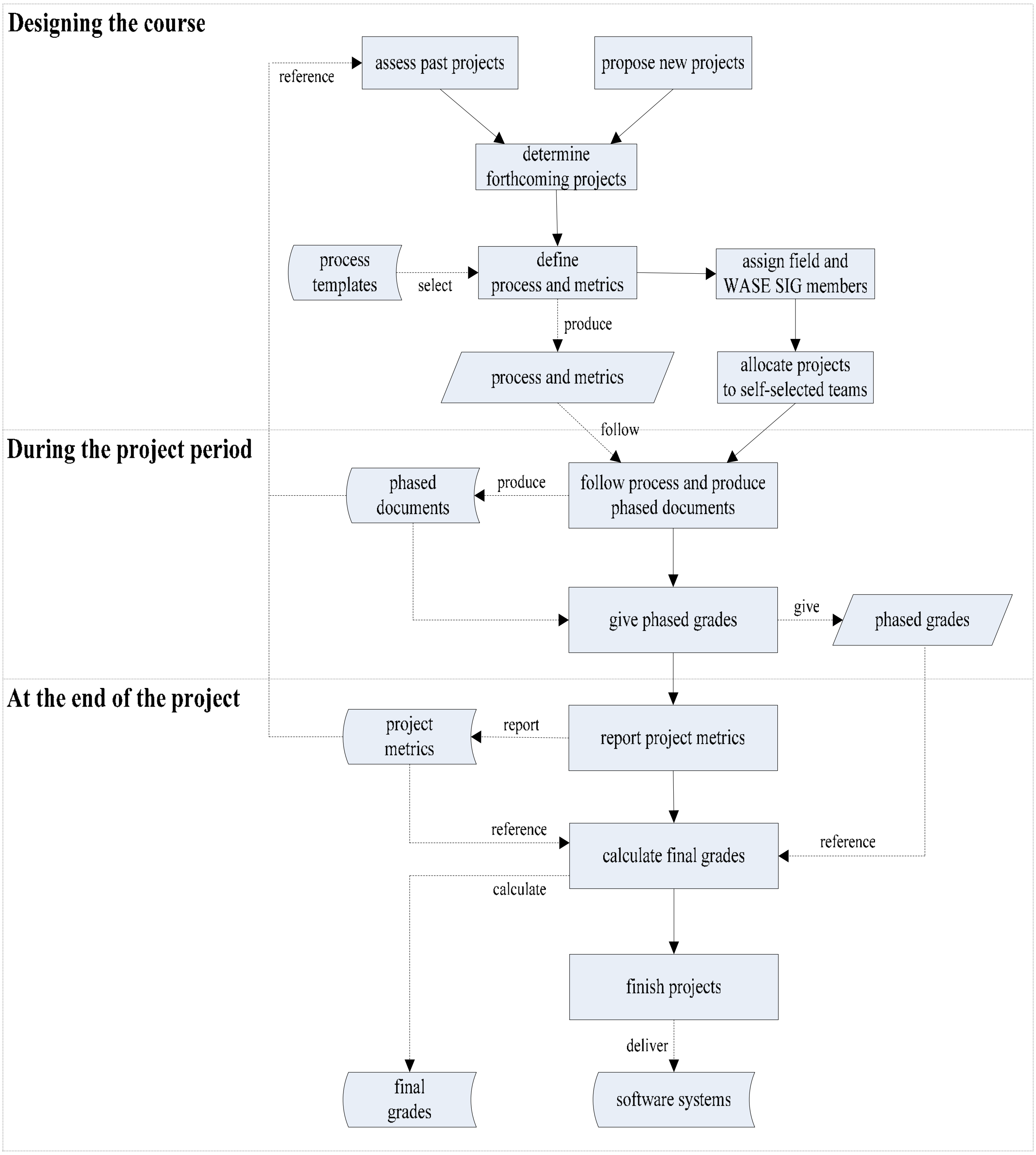

Figure 2: The Procedure of The Course Work Model

Designing the course

(1) SIGs meet together to assess past projects and their deliverables to determine whether any of them need to be enhanced or re- engineered for achieving better commercial realities.

This is an important step before determining the forthcoming projects. Since the software systems produced by past projects may not be complete or error-free, it is a good idea to enhance them for achieving better commercial realities to alleviate the fourth problem stated previously.

(2) The SIGs meeting determines the topics and contexts of the forthcoming projects.

In this phase, each SIG (except for WASE) may propose new projects ideas for its research or application purposes. However the SIGs meeting bases its decisions on both of those new ideas and the requirements of enhancing /re-engineering past projects. The final decision is always made to try to keep balance between SIG needs and problem alleviation.

(3) For each determined project, based on its subject and context, the WASE SIG defines a process and metrics to be followed and collected during the project period.

In this step, the SIGs meeting would ask the WASE SIG to define a process and metrics that are most appropriate to the project. The process is critical for the project to be successful, so the WASE SIG will assign a member to assist the team to follow it during the project period. Further, the metrics are collected for assessing the quality of the project and its deliverables. These metrics are useful not only for supervisors to grade the project and but also for the SIGs meeting to determine whether the project needs to be enhanced in the subsequent year.

(a) The WASE SIG defines some process templates for various kinds of projects and all of these templates are required to be kept at the maturity level three in terms of the academic maturity levels defined by [7]. Currently our processes defined are based on the Unified Process [8] with UML [9,10] and the SDLC process with iterative flavour [11]. These two well-known processes and their variants are sufficient for the development of various kinds of software systems. In addition, for maintenance or re-engineering of existing software systems, the WASE SIG adopts those processes defined in [12]. To comply with the standards in industry in Taiwan, all phased deliverables follow those formats specified by the Institute of Information Industry in [13].

(b) The WASE SIG defines a set of metrics to be collected during the project period. The metrics include (1) the percentage of finished out of desired functions, (2) the number of defects in finished functions, (3) how users satisfy finished functions, (4) how users satisfy the delivered system as a whole, (5) how useful users feel about the delivered system, (6) the number of person-hour spent in each project phase, and (7) the quality of the deliverables in each project phase. Among them, the first five metrics are collected at the end of the project, while the last two are collected in each project phase. These metrics are useful for supervisors to grade the project. In particular, the first five metrics are valuable as well for the SIGs meeting to determine whether the delivered system needs to be enhanced in the subsequent year.

(4) For each project, based on its field, the SIGs meeting assigns a field SIG member to supervise it. To facilitate the guidance of following the process and collecting metrics, a WASE SIG member is assigned to co-supervise the project.

(5) For each project, the SIGs meeting allocates it to a self-selected team according to the wishes of students. Since there are various kinds of

projects to be allocated, we always allocate them to students as complying with their wishes as possible. After the project has been allocated, the team proceeds to accomplish it within two semesters.

During the project period

(6) The project team follows the predefined process and produces phased documents before the deadlines for each phase. In addition, the team reports the number of person-hour spent for each phase. In each phase, the team may get field/WASE assistances from its twin supervisors by regular meetings, on- line discussions, or class lectures.

(7) At the deadlines for each phase, the project team turns in written documents to its supervisors. The deadlines are the project milestones, so these documents are reviewed by the supervisors to assess their quality. Based on the quality of these documents with respect to the number of person-hour spent by the team, the supervisors give grading for each phase. In our policy, supervisors give the grade with a maximum of five points for each phase.

At the end of the project

(8) The project team reports to supervisors those predefined metrics: (a) the percentage of finished out of desired functions, (b) the number of defects in finished functions, (c) how users satisfy finished functions, (d) how users satisfy the delivered system as a whole, and (e) how useful users feel about the delivered system.

The team collects the last four metrics throughout extensive tests: the number of defects in finished functions are those revealed by unit and integration tests [14,15] but still unsolved at this time, while the remaining three metrics for users satisfaction are collected via acceptance tests [14,15]. In collecting each of the last three metrics, users give quantitative grades with a maximum of five points and the total grade is divided by the number of users to derive the average grade for the metric.

(9) With the grades for each phase and those end-project metrics, the supervisors calculate the final grade for the team. The calculation is based on (1) the grade averaged from all phases, (2) the percentage of finished out of desired functions, (3) the number of defects per finished function, and (4) those grades for users satisfaction.

(10) The supervisors return all project materials including phased documents, delivered software system, collected metrics, and grades. These materials are put into a repository for the SIGs meeting to assess in the subsequent year. As specified in step 1, the SIGs meeting will review these materials to determine whether the project is successful or needs to be enhanced in the subsequent year.

4 Experiences of the model

In the past two years, we have experimented this model with three SIGs (EC, EL, and WASE) and five projects (three development projects proposed from respective SIGs where two of them derive further successive enhancement ones). We totally delivered three useless and two useful software systems. The useless systems were built up by project teams within two semesters, but not revised in the subsequent year. Their problems include unfinished functions, low grades for users satisfaction, and defects in finished functions. The two useful systems, a conference assistant system for our own department [16] and a wedding cloths management system [17], were developed by revising and enhancing two useless systems through successive projects to achieve better usefulness.

From our experiments, we find that the goals of our model as described in Section 1 have been met. That is, our model supports enhancing the software engineering skills of students and their experiencing effects, provides faculty with effective project administration to improve the quality of projects, and engages in continuous enhancements on software systems to achieve better commercial realities. With twin supervisors, students can get the assistance on technical as well as software engineering issues. At the SIGs meeting, software project processes have been formally defined and WASE SIG members are assigned to monitor whether they are strictly followed. Thus, the process maturity of our department can be easier maintained in a consistent manner around projects; supervisors have clear ways to administrate the projects and students get real experiences in applying sophisticated processes in the projects. This significantly improves the quality of projects and enhances the experiencing effects of students. In addition, the SIGs meeting also reviews past projects to determine if they need to be enhanced further for achieving better commercial realities. We find that this mechanism makes us not only deliver possibly useful software systems but as well offer certain portion of maintenance or re-engineering projects for students.

Our experiences also find that although our model supports very effective project administration, students are still probably unable to develop useful software systems from scratch within a project period. In our observation, this ascribes to two major factors: (1) they are not familiar with the application domain; and (2) they have no real experiences in the software and hardware utilized for developing the domain-specific application. In addition to taking regular courses, they need to spend lots of time on these two aspects before starting the development work. However, after we reviewed the useless software systems and devised successive projects to enhance them, we find that, by carefully practicing these systems and examining existing documents, students can debug out those problems around these systems and then successfully revise/enhance them to become really useful within the successive projects. After completing these successive projects, students involved appreciate their work much more than those joining only the development work from scratch: firstly, their work resulted in really useful systems; secondly, they learned more software engineering issues, not only development but also maintenance and re-engineering. These two advantages are not sensible to other project teams.

In contrast to the benefits from our model, there are some negative points we need to overcome. The first problem, possibly the most serious one, is that WASE SIG members are assigned to supervise all projects and hence their substantial amount of efforts on the supervision is not negligible. To alleviate their burden, we particularly reduce their teaching load on regular undergraduate courses. This is however not a way to solve completely the problem; in the long term, we hope to recruit more faculty members with software engineering expertise. The second problem is that the two supervisors for a project may have contradictions between them, maybe on administration styles, schedules, or attitudes. This disturbs both of supervisors and team members. We are now trying to get a better way to solve it, for example making two faculty members to work together with similar humanity or general interests.

5 Conclusions and Future Work

To address the problems in our current projects course work, we proposed a model of enhancing the management approach and presented in this paper. It utilizes the concept of SIG to assign field and WASE experts as twin supervisors for each project. Students can benefit largely from them to enhance their respective skills. In addition, software project processes have been formally defined at the departmental level; our department can maintain a sufficient level of process maturity. Thus, supervisors can administrate effectively the projects and students can get real experiences in applying sophisticated processes in the projects. The goals of improving the quality of projects and enhancing the experiencing effects of students have been met. In addition, by formally reviewing past projects, we can always offer certain portion of maintenance or re-engineering projects for students. These two subjects are very important but often not equally stressed in regular software engineering courses [18].

With our model, almost all of the students involved consent on the values of the projects course. They are also proud of their achievements in delivering software systems under the discipline of software engineering practices. In summary, our experiments show that our model is successful to improve the course work significantly. Hereafter, we plan to pervade our model to all departmental faculty and students. Since our model requires extensive management from field and WASE SIG members, some ways to reduce their administrative burden are necessary. One effective approach is to utilize the Web and Internet technologies to automate some administrative work. The Web site may provide supervisors with such services as giving comments and grades for phased deliverables, publishing related documents and lecture notes, and reserving meeting times [19]. We are now developing these Web services by proposing them as a software project in the projects course. The project is now supervised by the way we described in this paper. Its process adopted is the Unified Process with UML. The tools used include MS InterDev, ASP, VB Script, SQL Server, and some multimedia ones like PhotoShop, PhotoImpact, and CoreDraw. As stated earlier, we do not expect the Web site will be truly useful through this project. After it is finished in the next year, we shall have probabilities to revise and enhance it by a successive project.

References:

[1] E. Chang, T. Torabi, W. Rahayu, “Issues and Solutions for Running a Full Year Software Engineering Project for Computing Majors,” ICSE 1998, IEEE Computer Society pp. 106-112.

[2] B. Demuth, M. Fischer, and H. Hussmann, “Experience in Early and Late Software Engineering Project Courses,” the 15th Conference on Software Engineering Education and Training (CSEET 2002), IEEE Computer Society, pp. 241-248.

[3] A. Dutoit, B. Bruegge, and R. Coyne, “Using an issue-based model in a team-based software engineering course,” ICSE 1996, IEEE Computer Society pp. 130-137.

[4] L. van der Duim, J. Anderson, and M. Sinnema, “Good practices for Educational Software Engineering Projects,” ICSE 2007, IEEE Computer Society, pp. 698-707.

[5] J. Zhang, D. Zage, and W. Zage, “Improving Project Planning/Tracking for Student Software Engineering Projects through SOPPTS,” the 16th Conference on Software Engineering Education and Training (CSEET 2003), IEEE Computer Society, 2003.

[6] Web page of the Dept. of Info. Mgt. at MCU, http://www.im.mcu.edu.tw.

[7] J. Collofello, et al. (1994), “Assessing the Software Process Maturity of Software Engineering Courses,” Proc. of ACM SIGSCE ’94, pp. 16-20.

[8] J. Arlow and I. Neustadt (2005), *UML 2 and the Unified Process : Practical Object-Oriented Analysis and Design*, 2nd Ed., Addison Wesley.

[9] J. Rumbaugh, I. Jacobson, G. Booch (2004), *The Unified Modeling Language Reference Manual*, 2nd Ed., Addison Wesley.

[10] G. Booch, I. Jacobson, J. Rumbaugh (2005), *The Unified Modeling Language User Guide*, 2nd Ed., Addison Wesley.

[11] G. Shelly, et al. (2005), *Systems Analysis and Design*, 6rd Ed., Thomson.

[12] M. El-Ramly (2006), “Experience in teach- ing a software reengineering course,” ICSE 2006, IEEE Computer Society, pp. 699 – 702.

[13] Institute of Information Industry, *Software Development Guide*, 2003, http://www.iii.org.tw.

[14] R. Pressman (2005), *Software Engineering: A Practitioner’s Approach*, 6th Ed., McGraw Hill.

[15] I. Sommerville (2007), *Software Engineering*, 8th Ed., Addison Wesley.

[16] Home page of the Conference Assistant System of the Dept. of Info. Management at MCU, http://icim2007.mcu.edu.tw/system/logon.aspx.

[17] J. Lin, et al. (2006), “WebSphere Application Development - A Wedding Cloths Management System.,” Proc. of 2006 Info. Mgt. and Police Admin. Info. Conference, Taiwan, pp. 236-242.

[18] Chang Liu (2005), “Education & training track: Enriching software engineering courses with service-learning projects and the open-source approach,” ICSE 2005, IEEE Computer Society, pp. 613 – 614.

[19] K. Reid and G. Wilson (2007), “DrProject: a software project management portal to meet educational needs,” ACM SIGCSE 2007, vol. 39, issue 1, pp. 317- 321.