A Hybrid Learning Course on Software Development — Requirements Validation of Tool Support[1]

Y.T. Yu[2*], M.Y. Choy[*], E.Y.K. Chan[#], and Y.T. Lo[#]

Department of Computer Science, City University of Hong Kong
[*]{csytyu, csmchoy}@cityu.edu.hk, [#]{chanyk, ytlo}@cs.cityu.edu.hk

Abstract. Learning of software development demands not only adequate supervision by the instructor, but also intensive interactions among students. In traditional classroom learning, the number of contact hours between the instructor and students is very limited. This severely restricts the amount of guidance and learning that students may receive in a course. In particular, the best practices in software development, such as design modelling, peer review, quality assurance and project management, all require ample practice that is hardly feasible in the traditional classroom learning setting. Supported by e-learning systems and tools, a large part of the interactions between instructors and students can now be done online. We propose a hybrid learning design of software development courses to take advantage of both the rich context available in classroom learning and the benefits of electronic communications. This paper presents the rationale for hybrid learning in such courses, and describes a pilot hybrid learning course on software development for preliminary evaluation and requirements validation of tool support.

Keywords: Course design, hybrid learning, requirements validation, software development, Web-based learning tool.

1 Introduction

Software development (SD) courses aim at educating students the theories, techniques, and best practices of the development of software systems. As such, project work is an integral component of SD courses [15]. Development of non-trivial software projects has to be done in teams. In doing the project work, members of a team have to collaborate closely, as effective communication is one of the critical success factors in software projects. Moreover, the instructor has to offer adequate supervision to project teams and monitor the progress of students. Thus, intensive communication among students and the instructor plays an important part in software project work.

[1] This work is supported in part by a grant (project number: CityU123206) from the Research Grants Council of the Hong Kong Special Administrative Region, China.

[2] Corresponding author.

The importance of communication is particularly prominent from the perspective of learning in SD courses. First, students have to learn from the instructor how to plan a project and carry out the SD activities, which typically include requirements specification, design, implementation, testing and other quality assurance tasks. In this regard, ample guidance from the instructor is essential. Second, students have to learn from other members of the team through their collaboration in the project. During the project work, students invariably have to seek for supplementary learning materials which have to be shared among themselves. Third, modern SD best practices emphasise peer reviews and inspections, in which team members review the work of one another so as to remove any defects in the project deliverable as early as possible [2, 16]. Finally, members of a team can learn from the good work of other teams, provided there are opportunities for experience sharing and across-team reviews.

However, the number of face-to-face (F2F) contact hours between the instructor and students is usually very limited. This severely restricts the amount of guidance and learning that students may receive in a course. In particular, the best practices in software development, such as design modelling, quality assurance and project management, all require ample practice that is hardly feasible in a traditional classroom learning setting.

Web-based communications are more convenient, efficient, and flexible than F2F interactions as the former can be asynchronous and independent of the physical locations of the participants. Moreover, people can retrieve instant information from the Web and acquire new knowledge through it. Thus, the Web has opened up tremendous opportunities for improving the way that learning takes place.

Supported by e-learning systems, a large part of the interactions between instructors and students can now be done online. *Hybrid learning* [8, 12], also called *blended learning* [1, 6, 15], refers to the mode of education that requires the instructor and students to meet and interact not only in a traditional F2F classroom environment, but also online, typically through Web-based communication channels.

We have earlier proposed an outline of a hybrid learning design of SD courses [13], which takes advantage of both the rich context available in classroom learning and the benefits of communication by the electronic means. To facilitate the implementation of hybrid learning, we have built a Web-based course tool, known as TREASURE, to supplement an existing e-learning platform to provide the specific needs for learning of SD [14, 15]. For the purpose of preliminary evaluation and requirements validation of the tool support, we have recently completed a pilot run of a SD course based on the hybrid learning design and the use of TREASURE. This paper presents the rationale for hybrid learning in SD courses, and describes the learning activities in the pilot run.

The rest of this paper is organized as follows. Section 2 discusses the specific problems in conventional classroom learning in SD courses, and explains why it should be supplemented but not replaced by e-learning. Section 3 outlines the requirements, functionalities and design of TREASURE, which was custom-built for supporting hybrid learning in SD courses. Section 4 describes a pilot hybrid learning course on SD for preliminary evaluation and requirements validation of tool support. Section 5 briefly describes related work, and Section 6 concludes this paper with suggestions of further work.

2 Learning in Software Development Courses

2.1 Conventional Classroom Learning

The quality of software has become increasingly prominent since huge, complicated, and safety-critical software systems are now ubiquitous, affecting us in a myriad of ways in our daily life. To achieve high quality software, the SD process must be properly managed and well-disciplined. One common way of managing the SD process is to organise it into phases. A software process model is specified by the definition and sequencing of activities in these phases, together with the interactions among them. The most renowned software process model is the classical waterfall model, which is typically composed of a requirements definition phase, an analysis and design phase, a coding phase, a testing phase, and an operation phase. The waterfall model, which offers a structured approach to SD, provides distinct milestones and well-defined documentation in each phase of the process. It is perhaps for this reason that the waterfall model is commonly adopted in many SD courses [15], and also actually in most industrial SD projects [9, 10]. Here we present our ideas in terms of the waterfall model, but in fact the proposed course design can be implemented with the use of other process models, such as Boehm's spiral model and the agile processes [10].

In a typical SD course, the instructor has to form student groups, create software projects, allocate projects to different student groups, define the project phases, and prepare the document templates for students to record their intermediate and final work. When the work in each phase is completed, students should submit their intermediate deliverables to the instructor for assessment and feedback. The intermediate deliverables typically include requirement specifications, design models, test plans, progress logs and quality assurance reports [15, 16]. The performance of students should therefore be monitored and evaluated by assessing the quality of students' intermediate deliverables.

Learning of SD is communication and collaboration intensive. As each group works on the project, peer learning and review of intermediate work should be encouraged in order to maximise students' learning experiences. Thus, each group should share their work for comments by other groups. Fig. 1 depicts the flow of the tasks normally done by the instructor and students in a SD course [15].

Typically, tutorial sessions can be arranged so that students may discuss their work with other groups. However, such an arrangement has become increasingly difficult due to the limited F2F contact time, high student-to-instructor ratio and rigid class scheduling. Time and resource limitations often hinder the learning and teaching progress as well as the motivation of students in SD courses. Collaborative learning activities that are desirable beyond lectures and tutorials, such as in-depth group discussion, skills and experience sharing sessions, and technical information exchange, are also highly constrained when carried out in F2F settings. As such, students often receive little and/or delayed feedback from the instructor and fellow students. Moreover, this phenomenon is dissonant to the quality-centric notion that is advocated in SD courses and the industry's recommended best practices [2, 16].

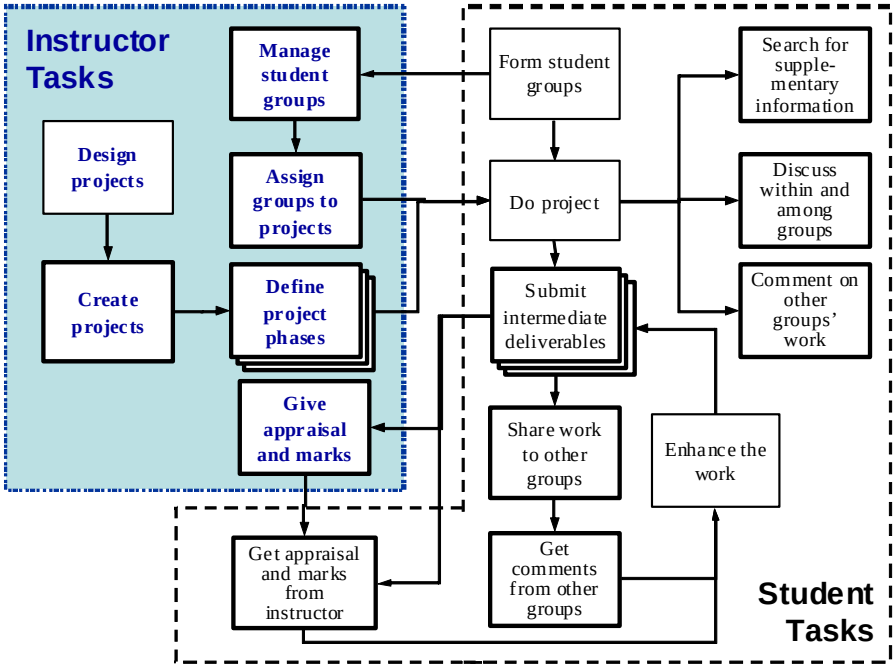

Fig. 1. Tasks to be done by instructors and students in a SD course [15]

2.2 E-Learning

More and more organizations begin to use e-learning as the major form of educational delivery, including universities, corporations, military institutions, and secondary schools [8]. On-demand availability of e-learning complements the routine structures of traditional classes by allowing students to participate and complete their coursework with flexible schedules in accordance with their daily family or work commitments, enabling more engaging learning materials to be accessible for a range of abilities and preferred learning styles, encouraging the development of an independent learning culture by making learning more inviting, and providing unique opportunities for active feedback from different participants. These benefits are particularly pertinent to our university in which the student population has a diverse background and different study modes, including the part-time evening mode [1, 16].

However, there are still values in the traditional classroom learning environment. In a typical SD course, the instructor has to offer guidance to students, encourage students to raise questions, provide comments and feedback, and act as the moderator to keep students on track. The classroom setting can promote social and cultural interactions among the instructor and students, facilitate mutual understanding through non-verbal communication mechanisms such as body language and eye-

contact, and encourage peer-to-peer learning. As such, a hybrid mode of learning can be much more effective than pure F2F learning or e-learning alone [14].

3 TREASURE: A Tool for Software Development Courses

Presently, a variety of course management systems are widely available, such as WebCT, Blackboard, and Moodle, and are commonly used in educational institutions. Through such a system, students can retrieve course learning materials, such as lecture notes, recommended readings, quizzes, assignments, assessment records, and surveys. The instructor can also make announcements to alert students whenever there are updated course learning materials or special arrangements in the forthcoming classes. Moreover, the instructor can create assignments to be completed by students who can later retrieve their results from the assessment database. Discussion boards provide facilities for online discussions among the instructor and students, enabling them to exchange information and share experiences.

However, these course management systems are not specifically designed for SD courses. They fall short of facilities for software project management such as managing student groups on a project basis and assigning projects to student groups. Peer review of software project work is not conveniently supported in these general course management systems. In order to better leverage the e-learning benefits, we have developed a Web-based tool, called TREASURE, to supplement our university's standard e-learning platform, Blackboard [14, 15].

TREASURE is built to facilitate the interactions needed for SD courses. It has to satisfy two major types of requirements for interactions, namely, Group Management and Project Management. Through the Group Management functions, the instructor can keep track of the membership of all the student groups, as well as the assignment of projects to groups. The Project Management functions allow the instructor to define new projects and their phases, enable within-group discussions and information sharing, as well as peer inter-group reviews, and provide facilities for the instructor's appraisal and assessment of the intermediate or final products of the projects.

To make the best use of the existing e-learning system, we design TREASURE as a plug-in module to Blackboard. TREASURE utilizes many of the Blackboard system's built-in databases and facilities, including (1) the Student Groups database for group management, (2) the Grade Book database for assessment, (3) the Content Folder database for storing learning materials, and (4) Forum for students to hold online discussions.

We use Java Server Page (JSP) to implement the Web pages of TREASURE. Teachers and students access different pages according to their roles in Blackboard. To satisfy the Group Management requirements, we use the built-in Blackboard APIs to access the basic group management features provided by Blackboard. Our tool provides a single and easier-to-use interface for teachers to add, modify and delete student groups, and to enroll students to groups in a batch or manually one by one. A new Java class for managing an external database of all project information is created to satisfy the Project Management requirements. In this way, the contents of the Project database will not affect the integrity of Blackboard's internal system data.

Throughout the requirements validation and other stages of the development of TREASURE, all stakeholders [4] (including instructors and students) are invited to provide feedback to its intermediate versions to validate the requirements and functions. All these activities are instrumental in ensuring that TREASURE can be usefully incorporated into our proposed hybrid learning approach for SD courses [15].

4 A Pilot Hybrid Learning Course on Software Development

In this section, we describe a pilot postgraduate SD course for preliminary evaluation of the hybrid learning design. The course relies heavily on TREASURE to provide the e-learning facilities that support the hybrid learning implementation. We shall first outline the course and its project work, and then describe in detail the learning activities and the use of TREASURE in satisfying the requirements of these activities.

4.1 The Course and the Project

Software Quality Engineering is a course offered to part-time students in the MSc Computer Science programme. Most students are software practitioners with 1 to 5 years of working experience. One of the course objectives is to enable students to develop and apply a working knowledge of good management and engineering practices for the production of high quality software. The coursework component requires students to work in teams on a project for the development of part of a real software system, with emphasis on using the methods of software inspection, peer technical review and independent verification and validation (IV&V) as the means of effective software quality engineering.

The project requires students to collaborate with their teammates in analyzing and verifying the requirements, designing the architecture as well as implementing a prototype of the software system. The project is divided into phases. At each phase, the instructor needs to monitor the progress of the project groups, and provide supervision and feedback to students. In addition, students have to review and comment on other groups' work at some phases of the project for the purpose of IV&V. The project work includes individual work done by students at their own time, group work done via F2F meetings in tutorial classes, as well as work done via communications and interactions in TREASURE.

4.2 The Course Learning Activities

At the beginning of the course, students form project groups by themselves and notify the course instructor. The instructor then manages the project groups' details by using the group management functions provided by TREASURE (Fig. 2).

Before the project kicks off, the instructor has to design a project by defining the required phases and deliverables that students are required to submit. The instructor then assigns dedicated student groups to the projects and phases. In this pilot course,

all student groups are required to do the same project. In other courses, different student groups may be assigned to do different projects.

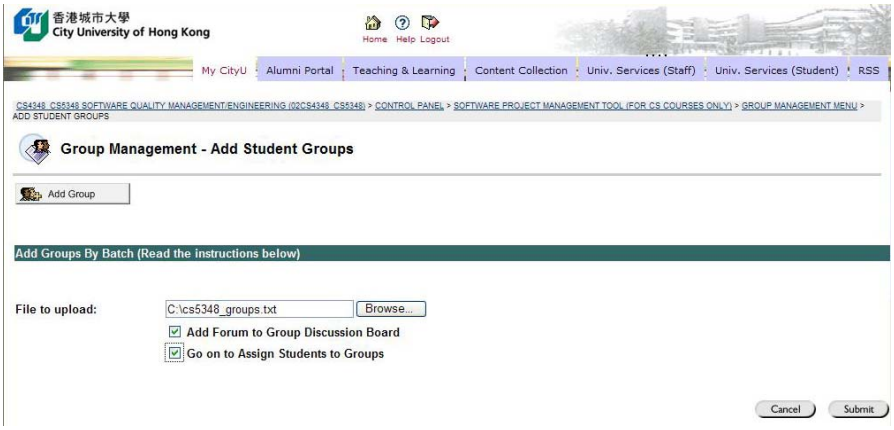

Fig. 2. Adding students to project groups

Once the project commences, students apply the knowledge they acquire in their study to do the project and to produce the required deliverables for various phases. Fig. 3 shows a snapshot of the use of TREASURE, which provides a single entry point for students to manage their project in a convenient manner. Students may collect the required reading materials, submit deliverables, communicate with one another and receive appraisal from the instructor through this interface.

We now describe the course learning activities in detail. Initially, a requirements specification of part of a real software system, seeded with a number of defects by the instructor, is given to students. Students apply the various SQE techniques learned in class to complete the project based on the given requirements specification.

The project proceeds in 5 phases. In Phase 1 Requirements Inspection, students are required to study and analyze the requirements specification individually using the perspective-based reading (PBR) method [11]. They are required to submit the individual potential defect list and the project plan as the first part of the Inspection Report at the end of this phase via the submission link in TREASURE (Fig. 3). While the defect detection can be done individually, students in the same group have to collectively decide a project plan. Since all students have full-time job and it is very difficult to arrange F2F meetings within the short time duration of this phase, they have to communicate electronically in planning the project. The instructor may also participate in the electronic discussions to monitor students' progress or help those who have difficulties in their project planning.

In Phase 2 Requirements IV&V, students have to arrange two formal inspection meetings to review the requirements specification and to compile an agreed defect list and a revised requirements specification to be included in the second part of their Inspection Report. The first inspection meeting is held by the group members themselves. During the meeting, each group follows the Fagan's inspection

process [5] as closely as practicable. The second meeting is held by a third party to perform IV&V. Both meetings are done in the classroom so that students can experience the F2F software quality assurance practice that takes place in real life.

In Phase 3, students work together to produce a detailed design document that includes system architecture and design modelled by data flow diagrams or UML diagrams, database tables, data structures as well as the algorithms of the system components. They have to surf the Web for supplementary learning resources. This kind of self-learning activities may apply to other project phases as well. To facilitate effective self-learning, we have compiled a set of useful Web resources for students' references. Fig. 4 shows the Web page of SD resources for the course.

After completing their work, students upload their draft intermediate deliverables to TREASURE for other groups to comment. In the mean time, the instructor may also comment on students' work drafts. Afterwards, each group can improve their work based on the comments they receive before formal submission for assessment.

Fig. 5 shows the interface of TREASURE through which students may view and comment on the work of other groups. In conventional classes, it may take a long time to complete this activity. To allow students to give F2F comments on the work of others, some precious tutorial class sessions may be needed. Similarly, it is also hard for the instructor to give appraisal promptly. Nor is it feasible for the instructor to give F2F comments to each student individually in class, as while talking to one student, other students would have to waste their time waiting for their turns in class. Without tool support, timely comments would be very difficult to implement.

At the completion of Phase 4 Prototype Development, students perform a F2F demonstration of their system prototype to other students and also review other students' draft prototype so as to provide comments for mutual improvement. The peer review of the system prototype is based on its conformance to the system requirements, consistency of the interface and usability of the system. The work in this phase is done F2F in tutorial classes. Similar to Fig. 5, there is another area in TREASURE for students to upload their comments in this phase.

In the final phase, students revise their draft system prototype according to the comments from the instructor and other students. Meanwhile, they have to submit an acceptance test plan together with a consolidated final report in which all relevant information about the prototype should be documented. Instructor can then mark the student submissions and give appraisals to each individual student via TREASURE.

5 Related Work

One work related to ours is the Web-based Collaboration Support Sub-system of an Education Support System developed in Tokyo Gakugei University [7]. The system allows instructors to act as inspectors to review and comment on the artifacts created by students. It also supports version and configuration management of the submitted artifacts. To monitor students' progress, the system keeps track of different states of the artifacts being inspected. A bulletin board is provided for discussion between individuals within group, among groups, and between groups and the instructor side.

Fig. 3. TREASURE allows students to manage their activities online

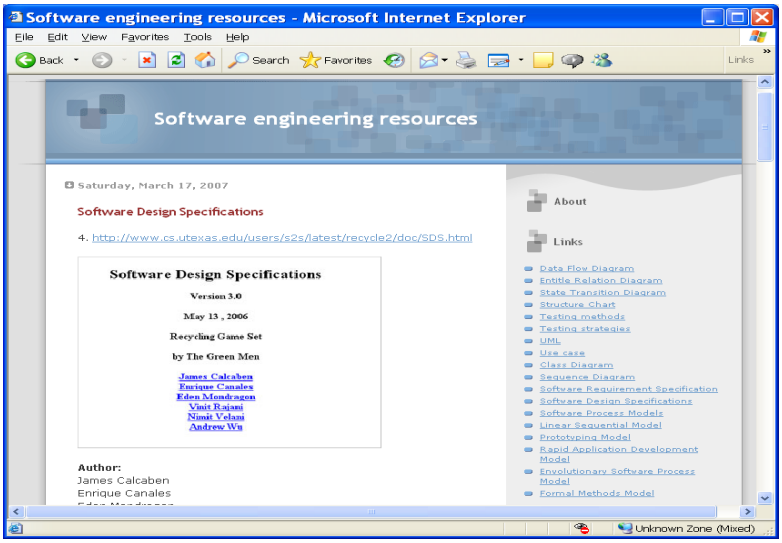

Fig. 4. Software design reference materials in software engineering resources blog [13]

Another related work is the ClassCompass system developed in the University of British Columbia as a distributed tool for group design mentoring [3]. It consists of several components: the ClassCompass Server, Instructor Client and Student Client. It provides a Web-based graphical editor for students to edit their UML diagrams. An automatic critique system generates expert advice when common design mistakes, such as association cycle and unnecessary realization, are found. Students can then revise their designs based on the generated advice. After submitting their own initial designs, students can critique on other groups' work, based on design principles that are pre-defined by the instructor. Instructors can act as experts to view the designs and manually provide feedback to students so that they may refine their design.

In short, the Web-based Collaboration Support Sub-system is designed to facilitate the inspection activities performed at the testing phase, and ClassCompass provides functionalities for mentoring the activities of students performed at the design phase. While each of these two learning support systems is specifically designed for a particular phase of the SD process, TREASURE realizes our hybrid learning approach at both the activity and course levels by assisting both the instructor and students in the entire SD life cycle. Secondly, the two learning support systems are standalone systems, whereas TREASURE is a plug-in tool built on the Blackboard architecture to take advantage of the functions of the existing e-learning platform. Thirdly, while the two learning support systems do not provide any functionalities for project group management, TREASURE supports the formation of project groups for different projects. As a plug-in tool, TREASURE requires less learning effort than if built otherwise, because students are already familiar with the user interface elements and user interaction metaphors of the Blackboard system.

Finally, even though many commercial tools, such as Microsoft Project, can facilitate software project management, they are not designed for teaching and

learning in SD courses. There are also tools (such as Rational Rose) designed for computer-aided software engineering, but they do not have appraisal functions. TREASURE is specifically built for the purpose of teaching, learning and assessment in SD courses, with the expectation of realizing our hybrid learning course design.

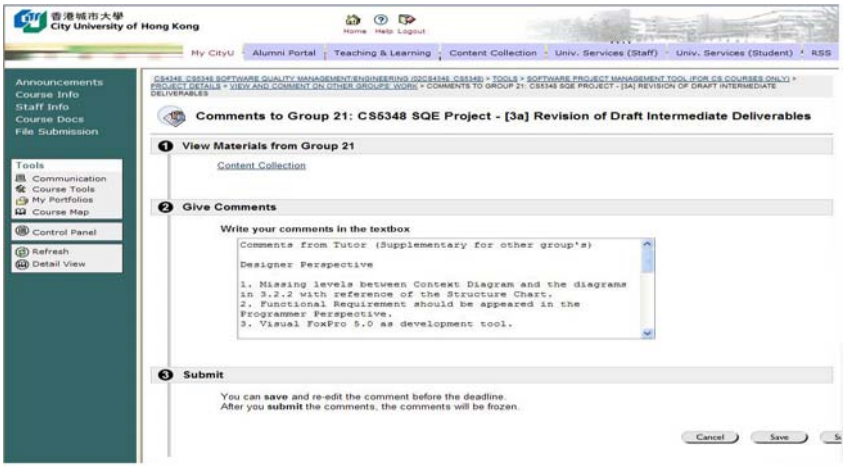

Fig. 5. View and comment on other groups' work

6 Conclusion

In a conventional classroom, many factors hinder learning in a SD course, such as the limitation of time for interaction among participants, resource constraints such as high student-to-instructor ratio, and the diversity of students' backgrounds and study modes. A SD course requires students to do group project work to practise software engineering principles and proven techniques. Timely feedback and regular peer reviews are essential not only to ensure the quality of the product, but also to enhance students' learning experiences. The use of asynchronous communication can help improve students' skill of team work and time management. All these factors call for the adoption of a hybrid approach combining classroom learning and e-learning.

This paper has described the rationale of using a hybrid approach in SD courses and the learning activities in a pilot run of such a course, which utilizes our custom-built tool, TREASURE, to organize a non-trivial software project for students to learn the essentials of software quality engineering practices. Looking ahead, it would be desirable to integrate TREASURE with students' SD environment as far as possible. Most intermediate deliverables or products of a software project such as design models and software prototypes have to be assessed for evaluating students' progress. Further work may be done to automate more parts of the assessment processes by integrating our tool with other project assessment tools.

Acknowledgment. We thank Roy Au for his work in developing TREASURE, and Alfred Chan for helping out in the course. A preliminary version of this paper was earlier presented at the Symposium on Hybrid Learning 2007 [13].

References

1. Choy, M., Lam, S., Poon, C.K., Wang, F.L., Yu, Y.T., Yuen, L.: Towards Blended Learning of Computer Programming Supported by an Automated System. In: Workshop on Blended Learning 2007, pp. 9–18, Prentice Hall (2007)
2. Ciolkowski, M., Laitenberger, O., Biffl, S.: Software Reviews: The State of the Practice. IEEE Software 20(6), 46–51 (2003)
3. Coelho, W., Murphy, G.: ClassCompass: A Software Design Mentoring System. ACM J. on Educational Resources in Computing 7(1), Article 2 (2007)
4. Damian, D.: Stakeholders in Global Requirements Engineering: Lessons Learned from Practice. IEEE Software 24(2), 21–27 (2007)
5. Fagan, M. E.: Design and Code Inspections to Reduce Errors in Program Development. IBM Systems Journal, 15(3), pp. 182–211 (1976)
6. Graham, C.R., Allen S., Ure, D.: Benefits and Challenges of Blended Learning Environments. In: Khosrow-Pour, M. (ed.) Encyclopedia of Information Science and Technology, pp. 253–259. Hershey, PA: Idea Group (2005)
7. Hazeyama, A., Nakako, A., Nakajima S., Osada, K.: Group Learning Support System for Software Engineering Education – Web-based Collaboration Support between the Teacher Side and the Student Groups –. In: First Asia-Pacific Conference on Web Intelligence: Research and Development. LNAI, vol. 2198, pp. 568–573. Springer-Verlag (2001)
8. Koohang, A., Durante, A.: Learner's Perceptions toward the Web-based Distance Learning Activities/Assignments Portion of an Undergraduate Hybrid Instructional Model. J. Inform. Tech. Edu. 2 (2003)
9. Laplante, P.A., Neill, C.J.: 'The Demise of the Waterfall Model Is Imminent' and Other Urban Myths. ACM Queue 10(1) (2004)
10. Neill, C.J., Laplante, P.A.: Requirements Engineering: The State of the Practice. IEEE Software 20(6), pp. 40–45 (2003)
11. Shull, F., Rus, I., Basili, V.: How Perspective-based Reading Can Improve Requirements Inspections. IEEE Computer 33(7), pp. 73–79 (2000)
12. Young, J.R.: 'Hybrid' Teaching Seeks to End the Divide between Traditional and Online Instruction. The Chronicle of Higher Education 48(28) (2002)
13. Yu, Y.T., Choy, M.Y., Chan, E.Y.K., Lo, Y.T.: Learning of Software Project Development: Towards a Hybrid Approach. In: Fong, J., Liu, L.C., Wang, F.L. (eds.) Hybrid Learning: Symposium on Hybrid Learning 2007, pp. 333–338 (2007)
14. Yu, Y.T., Choy, M.Y., Chan, E.Y.K., Lo, Y.T.: A Web-based Tool for Software Project Coursework: Requirements, Validation and Implementation. Presented at the 2007 International Conference on ICT in Teaching and Learning, Hong Kong (2007)
15. Yu, Y.T., Choy, M.Y., Chan, E.Y.K., Lo, Y.T.: Requirements and Design of a Web-based Tool for Supporting Blended Learning of Software Project Development". In: Hirashima, T., Hoppe, U., Young, S.S.-C. (eds.) Supporting Learning Flow through Integrative Technologies — Proceedings of the 15th International Conference on Computers in Education (ICCE 2007), pp. 159–166. IOS Press (2007)
16. Yu, Y.T., Poon, P.-L.: Designing Activities for Learning Software Quality Practices. In: 5th International Conference on Quality Software (QSIC 2005), pp. 333–338. IEEE Computer Society Press (2005)