



Software engineering education: A study on conducting collaborative senior project development

Chung-Yang Chen^{a,*}, P. Pete Chong^b

^a Department of Information Management, College of Management, National Central University, Taoyuan 300, Taiwan

^b College of Management and Design, Mingchi University of Technology, Taishan, Taipei, Taiwan

ARTICLE INFO

Article history:

Received 17 November 2009

Received in revised form 31 July 2010

Accepted 29 October 2010

Available online 10 November 2010

Keywords:

Software engineering education

Senior project

Collaborative development

Meetings-flow

ABSTRACT

Project and teamwork training is recognized as an important aspect in software engineering (SE) education. Senior projects, which often feature industrial involvement, serve the function of a 'capstone course' in SE curricula, by offering comprehensive training in collaborative software development. Given the characteristics of student team projects and the social aspects of software development, instructional issues in such a course must include: how to encourage teamwork, how to formalize and streamline stakeholder participation, and how to monitor students' work, as well as sustain their desired collaborative effort throughout the development. In this paper, we present an exploratory study which highlights a particular case and introduces the meetings-flow approach. In order to investigate how this approach could contribute to the project's results, we examined its quantitative benefits in relation to the development of the project. We also conducted focus group interviews to discuss the humanistic findings and educational effects pertaining to this approach.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

Recent publications indicate the importance of software engineering (SE) education and promote project and teamwork training in SE curricula (Mead, 2009; Rooji, 2009; Sancho-Thomas et al., 2009; van der Duim et al., 2007; van Vliet, 2006). Among the SE courses, senior projects, or so-called senior design projects, serve as a capstone course that allow students to work in teams and apply what they have learned during their college years to implement a computer-based system. The duration of senior projects varies from six months to a year, which is longer than normal curriculum coursework, and allows the students enough time to fully complete all aspects of the project.

Given the characteristics of student projects and how software development is carried out, collaboration and related training must be emphasized in senior project development. Students lack experience in, and knowledge of, the complete development of long term projects (Hassan, 2008; Chamillard and Braun, 2002); therefore, how to encourage and sustain students' teamwork throughout the development may be a challenge for instructors. Instructors must not only foster teamwork, but also monitor students' work against the team-related syndromes. Although more supervision

of the students 'in-process' would be helpful for the instructors, it would also be time-consuming.

In order for students to experience real problems rather than mere academic exercises, many schools welcome industrial involvement in senior projects. In these cases, students may develop prototypes or IT features for the sponsoring company. Many studies have promoted the industrial involvement in senior project development (Mead, 2009; Fornaro et al., 2007; Reichlmayr, 2006; Wohlin and orn Regnell, 1999; Kornecki et al., 1997; Scragg et al., 1994). However, such involvement, if not handled appropriately, may result in generating more problems than solutions. Some studies (van der Duim et al., 2007; van Vliet, 2006; Dawson, 2000) have argued that inexperienced students are not equipped to deal with the dirty tricks that can occur in the industrial world and their involvement may actually prove to be a counterproductive learning exercise. Furthermore, the instructor may have to bear the brunt of contractual responsibilities, as students would not have as much liability regarding the results of the project.

The external stakeholders may also be compromised in the involvement of industry in student projects. As the project is based upon the needs of the company, the ongoing participation of external stakeholders from the company is considered to be important (Reichlmayr, 2006; Sikkil et al., 1999; Mead et al., 1999). However, our experiences show that when meeting with students, external stakeholders often feel time is wasted because the discussion is often repeated questions on introductory issues. The agenda may be ill-prepared or may lack connections to the previous and future

* Corresponding author.

E-mail addresses: cychen@mgt.ncu.edu.tw (C.-Y. Chen), chong@mail.mit.edu.tw (P.P. Chong).

agendas, thus consequentially discouraging the participation. In considering these aspects, the educational issues as well as the motivations of this paper for effectively conducting a senior project and its collaborative development, are summarized as follows:

1. The need for a method to encourage and direct teamwork.
2. The need to formalize and streamline stakeholders' participation.
3. The need for an ongoing way to monitor students' work and to ensure that the desired collaborative effort is sustained throughout the development of the project.

In this paper, we present an exploratory case study of a distinguished senior project from the Computing–Engineering Undergraduate Program (CEUIM) at Taiwan's National Central University, and introduce the approach used to resolve these issues in the project. This work was selected as a distinguished project in the institute and later won the industry–university collaboration excellence award in the 2009 Taiwan National College Contest of Information Management Project Design and IT Services Innovation. This is an annual, nationwide senior project contest and attracts an entry of more than one hundred student teams from universities and colleges all over the country. To address the aforementioned issues, those involved used a specific approach that focused on *meetings*; particularly project meetings, which are often time-consuming and considered to be of little importance. A meetings-flow approach was used to effectively conduct the collaborative project development. Two types of meetings-flow were introduced by the case. One was the *temporal meetings-flow*. This promoted the means for pursuing ongoing substantial teamwork as well as gauging development. The other was the *contextual meetings-flow*, which helped to transform a student group project into a fluent execution of team collaboration and stakeholder participation.

In order to measure the effect of this approach on the results of the project, we conducted a quantitative investigation. The investigation also assessed the project and examined the numerical benefits that the approach brought to the project development. Finally, this study conducted focus group interviews to discuss the qualitative and educational effects of the approach on various project stakeholders, in order to consider the more humanistic aspects of software engineering. For the rest of this paper, it is organized as follows. Section 2 presents a survey of related literature. Section 3 presents the research method, the project case and the approach. The quantitative investigation is described in Section 4. The findings from the focus group analysis are presented in Section 5. Section 6 discusses the validation of the project case and analysis. Section 7 concludes the discussion and offers outlines for future research directions.

2. Literature review

2.1. Characteristics of senior projects

In this section, the characteristics of senior projects relevant to this paper are reviewed. The first characteristic is the changeable scope of a senior project. In contrast to employees, the typical student involved in a senior project does not have sufficient experience and is more susceptible to outside distractions (Abernethy et al., 2007; Dawson, 2000; Schlimmer et al., 1994). These two factors may contribute to an increased degree of unreliability, and the parameters of the project may need to be continually adjusted accordingly.

The second characteristic is team-orientation, which is also known as mutual-reliance in student project work (Rooji, 2009;

Hassan, 2008; Abernethy et al., 2007). According to Lipman (2003), the team nature of student projects promotes learner–learner interaction. However, because students lack experience, knowledge of the entire software development and relevant ability, they are more likely to simply explore (sometimes blindly) the project (Hassan, 2008; Chamillard and Braun, 2002). They need to have a plan and be given instruction to direct and sustain the required team effort throughout a software project's development.

The third characteristic of many senior projects is the involvement of external companies. According to empirical studies, software industries often complain that the training in software development in schools is strictly academic (Denning, 1992; Mead et al., 1999; Moore and Berry, 1999). Therefore, many schools welcome the involvement of external companies to enrich their training programs. This characteristic appears to be significant in senior project education, as a number of industrial-academic project reports and case studies are profiled in the literature, e.g., (Mead, 2009; Fornaro et al., 2007; Reichlmayr, 2006; Moore and Berry, 1999; Sikkel et al., 1999; Wohlin and orn Regnell, 1999).

The fourth characteristic refers to the risky nature of research orientation (Chamillard and Braun, 2002; Schlimmer et al., 1994). In many undergraduate programs, research-oriented senior projects are a stepping-stone to graduate studies. Hence, unlike real-world projects that are often bound by contracts, students have the luxury of trying out new ideas with relatively minor consequences. Should the industry-involved project fail, they may receive poor grades, but it is the instructor who may bear more severe consequences. Students, therefore, may not be as fully committed (Sancho-Thomas et al., 2009) to the project, or assume as much liability as do those in industry.

In their article “Teaching Teamwork”, Hilburn and Humphrey (2002) suggested a cyclic development (CD) strategy to manage student group project development. CD applies “divide-and-conquer” and “watch-and-go” strategies by focusing on a subset of requirements and implementing them within system functions. During each development loop, the types and levels of risks/uncertainties are monitored and a subset of requirements for the next cycle is selected. As this concept seems to fit the characteristics of the senior projects mentioned above, it becomes the basis of the approach in the reported case to further address the collaboration issues in this paper.

2.2. Collaboration training in senior projects

In software project development, collaboration plays a critical role due to the complicated product and people environment. Therefore, many studies advocate incorporating project-based teamwork training in SE curricula (Mead, 2009; Rooji, 2009; van der Duim et al., 2007; van Vliet, 2006; Hilburn and Humphrey, 2002). In this regard, the senior project course offers a project environment in which students can receive comprehensive training on collaborative software development.

Software development collaboration differs from the team-oriented nature of student projects mentioned in the previous section. Collaborative software development in senior projects may be internal (i.e., among team members) or external (i.e., with outside members such as the instructor or sponsoring company). Denning (1992) and Scragg et al. (1994) argued that computing education does not prepare students to work in joint development environments populated by people in various roles. They also pointed out that often the skills students lack include the ability to communicate and to collaborate with others, rather than technical skills. Many articles in the computing and engineering education literature have identified this gap and advocate educating students with more group-oriented collaboration training (Sancho-Thomas

et al., 2009; van der Duim et al., 2007; van Vliet, 2006; Melin and Cronholm, 2004; Hilburn and Humphrey, 2002; Favela and Peña-Mora, 2001).

In conducting students' internal collaboration, instructors may need to pay attention to three syndromes owing to students' team-oriented nature. The first of these is the "free-rider syndrome" (Goldratt, 1997). It is known that when students work as a team, because the group as a whole is being graded, not all of the students work equally hard. van der Duim et al. (2007) reported that this syndrome can be a major problem in student team project development. The second is "Wyatt Earp syndrome" (Boehm and Port, 2001), which may also occur in a student group project and hinder collaboration when a student presents a new idea or accomplishment to the instructor without having first discussed it with other members of the group. The third is the "student syndrome" (Marchewka, 2009; Goldratt, 1997), i.e., the phenomenon that many people do not fully apply themselves to a task until the last possible moment before a deadline. These three factors may exacerbate the uncertainty of senior projects mentioned earlier.

Melin and Cronholm (2004) derived similar findings and indicated the need to manage student projects by examining the processes of collaborative work. These issues are addressed in this paper; student work is monitored by an ongoing assessment of their collaborative work, in order to reduce the team-related syndromes.

In Mead's (2009) milestone review report on SE education and student project training, it was suggested that industry–university collaboration in software engineering education be continued in the future because such collaboration would result in joint research and keep the educational input grounded in practice. However, if not handled appropriately, industry-driven student projects may not achieve the desired training and education results. For example, Dawson (2000) presented twenty dirty tricks that should be provided or simulated in an industrial project, for helping prepare and train computing students to be practical and flexible in real business and human environments. When considering incorporating these tricks into an industrial project, van Vliet (2006, p. 56) pointed out that without a gentle and progressive guidance, these practices may not be helpful to SE students who are not mature and professional enough to handle the tricks. Students might spend much more effort in struggling with the real-world training than in preparing themselves to develop best practices and social skills in SE disciplines.

As well as the 'dirty-trick' overload for the students and the contractual burden for the instructor, another factor to be considered is that external stakeholders may be negatively affected by students' poor communication and inexperience when they participate in the development (van der Duim et al., 2007). In addition to lacking interpersonal communication skills mentioned in Fornaro et al. (2007), our experience has shown that students' inability to manage discussions lead to poor communication results. These poor and inconsistent communication skills either discourage the external stakeholders from meeting with students at all or less influential personnel are sent to participate in the discussions. If the stakeholders are to ensure the effective and continuing participation of industry in student projects, it is essential to formalize and streamline the discussions. The sensitive and careful intervention of the educator/instructor may be necessary to facilitate the student-company collaboration.

3. Methodology

3.1. Research and data collection method

To address and explore the "how" questions (Yin, 2008) that are raised in this paper, an in-depth case study of a distinguished senior project is presented. This study manifests the following three

characteristics: firstly, it is a case study rather than a controlled experiment. Secondly, participants include full-time senior students and software company stakeholders. Thirdly, the data are obtained from the participants' logbook. This journal had been required since the project began; participants were not rolling out the logbook for this case study.

The data collection and analysis in this case study was conducted for three purposes: (1) to present the project case and the approach, (2) to investigate the benefits of the approach for the project development, and (3) to identify and discuss the effects of the approach on project stakeholders. Data collection for the first purpose includes the background and motivational information of the case, as well as the observations from using the approach. The data used represent both direct and indirect evidence; the direct evidence comprises the first-level data and the logbook information regarding the execution of the method while the indirect evidence refers to the supplementary interviews that support the execution of this study. Direct evidence was also used in the analysis for the second purpose, which was to conduct hypothesis testing in order to examine how the approach contributed to the numerical outcome.

As to serve the third purpose, we took a qualitative method. In empirical software engineering and system development, qualitative methods and data are used in a case study to address non-technical issues, such as the human aspect (Seaman, 1999; Darke et al., 1998). In the 1950s, focus groups emerged as a research method in the social and educational fields. They have gradually become recognized as necessary research tools in SE studies, especially when humanistic aspects need to be studied (Kontio et al., 2004; Vaughn et al., 1996). According to Morgan (1996), this method is useful for identifying new subjects when a new approach or prototype for interviewing user groups is needed. It is often used when conducting interviews within existing groups since it offers more interactive discussion. We conducted focus-group interviews with the student team, the instructor and the company stakeholder who had all been working together for the eighteen weeks required for the project development. The aim of the interviews was to explore and report on the findings regarding the human-related and socio-technical aspects of the approach and the study.

3.2. Project description

The senior project under discussion is titled Project Issues Monitoring Information System (PIMIS); it is a project development from the Computing–Engineering Undergraduate Program (CEUIM) at Taiwan's National Central University. CEUIM serves as the SE education track in the institute, and its primary goal is to help students establish software development capabilities with a series of SE courses, including the senior project course. In order for students to experience real problems, rather than mere academic exercises, CEUIM welcomes industrial involvement in senior projects. The senior project course in CEUIM has two semesters (about 9 months). When the time taken for administration, introduction to the course and closure is deducted, the actual time available for students to develop the projects is about six months. Within this timeframe, students need to develop prototypes of innovative products or IT features for a sponsoring company. Although senior projects may be relatively small, the students find the course to be the most demanding.

The PIMIS senior project was sponsored by ESNE, a CMMI-compliant software company. CMMI (Capability Maturity Model Integration) is an industrial standard that software companies use to standardize software project development. Starting in early 2008, ESNE ran a long-term system development plan called "Mobilizing CMMI" for incrementally developing a CMMI-based project management system in a mobilized environment. The project in

this case study was part of the entire development; it implemented the PPQA, VER and PMC process areas of CMMI. The following section explores the project case and introduces the approach by first describing its motivational story.

3.3. Conducting collaborative PIMIS project development

3.3.1. Focusing on meetings

Software project development relies on human beings in all practical cases, including senior projects. Yet managing people is not an easy work, and is also a critical lesson in the PIMIS project since it depends heavily on teamwork and external stakeholder's participation. When there is a team or when people are involved, major problems are (The Standish Group, 2007; Marchewka, 2009), e.g., lack of user or stakeholder involvement, inadequate managerial support, ignorance of what other team members are doing, or students' team-oriented syndromes mentioned earlier. It seems paradoxical that these problems often arise despite the amount of time that the project members spend in meetings.

Meetings are intended to be beneficial. According to some studies, teamwork in meetings enables the participation and interaction of all people involved in a collaborative work and promotes the sharing of inspiration and knowledge (Hass, 2006; Gallivan and Keil, 2003; Wenger et al., 2002). Meetings offer a more formal environment for stakeholders involved in the project. Based on this, the PIMIS instructor decided to use meetings to encourage teamwork and formalize stakeholder participation.

However, it became apparent that there were problems with the structures and practices already in situ. As the project evolved, people seemed to spend a great deal of time and effort in numerous scheduled or unscheduled meetings but the results often seemed to be ineffective. Aside from administrative and facility problems, known issues are that suggestions made during the meeting were not acted upon, or follow-up checks were not properly undertaken. People often focus on local, current issues and do not take a more global view regarding project development. There is a lack of insight with regard to long-term planning and the contextual relationships between previous, current and future meetings. In other words, current meeting practice does not have a global view of meetings that happen during project development; nor has there been a way to define and manage project meetings and further integrate them into a project process to yield a more effective managerial result.

The PIMIS project reconsidered the often ignored, yet time-consuming project meetings, and developed a meetings-flow (MF) approach, whose aim was to help in the instruction of student teamwork and to formalize stakeholder participation for the duration of the project. In the following subsections, an in-depth, technical description of the approach is provided. Special attention is given to the establishment and execution of the meetings-flow in the project case.

3.3.2. The meetings-flow approach

The concept of the flow of meetings was seen in Chen's (2009) preliminary work presented in a computing educational conference. This study extended the concept to fully develop the meetings-flow approach. Here, *meetings-flow* is not about the micro-level of internal control flow within a meeting; instead, it refers to a holistic modeling of substantial teamwork of collaborative development into various functional meetings and the interconnectivities among the meetings. In the PIMIS project, the approach comprised of three steps: (1) identifying a cyclic development lifecycle model and the corresponding work breakdown structure (WBS), (2) defining student-centered functional meeting classes based on the WBS, and (3) organizing the meetings to form the meetings-flow (MF).

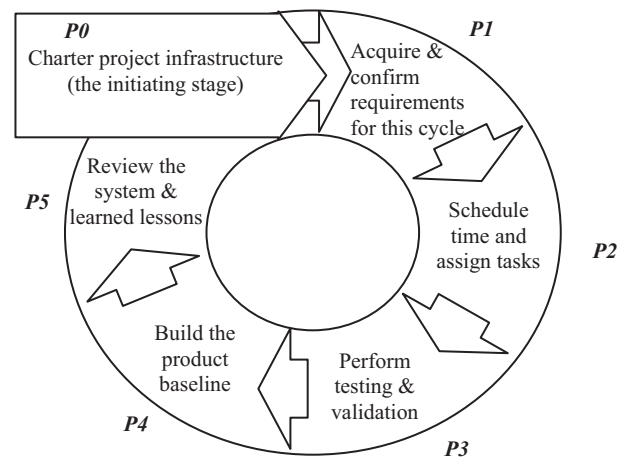


Fig. 1. The cyclic SDLC development model in the PIMIS project.

3.3.2.1. *Identify the development model and the WBS.* Because senior projects carry risks and can be unstable, the PIMIS senior project was based on the cyclic development strategy (Hilburn and Humphrey, 2002). The corresponding WBS was planned as the project's development loop. Fig. 1 illustrates the proposed development model with the relevant work items shown in each development loop. It can be seen that the PIMIS team first set up the charter of the project, including the main goal, scope and total time-frame, and then broke the project down into smaller components, using the cycle.

Two types of WBS: procedure-oriented and functional, were used to develop the PIMIS product. The upper two levels of WBS refer to the system modules and sets of related functions which are grouped as a development batch. The lower level of the WBS refers to the cyclic development tasks for implementing the function groupings one by one. In the PIMIS case, there were about 21 development cycles, allowing one or two weeks per cycle. The system development would be accomplished in under six months.

With the development of the WBS, meeting classes were required to identify and establish how the project development was to be implemented and who was to participate in the various activities. However, as there were too many group activities during the development, it became imperative to identify which were essential and thus mandatory for the stakeholders.

3.3.2.2. *Define the student-centered functional meeting classes.* In communication action research, the 5W1H theory is a structural method to characterize organizational communication (Yoshioka et al., 2001). In the PIMIS project, organizational communication could be applied to the situations in which students meet in their groups or when external stakeholders participate in the students' project work. These situations were defined in the project as "planned collaborative or participative functions" (coded as CP functions). In these contexts, CP functions were characterized by *who* performs *what* CP functions with/to *whom*. For example, the instructor (*who*) has a participative function (*what*) entitled "project review" for a student team (*whom*).

By regarding stakeholders as classes, the instructor applied the CRC (Class Responsibility Collaboration) method (Beck and Ward, 1989; Brown, 2002) and conducted a role-playing game to identify substantial CP functions. In this project, stakeholders included the five-student project team, the instructor and the representative from ESNE. The student team was further divided into two sub groups: one for leading system implementation and the other for hosting the R&D issues of the project. During this game, the students played the roles of the instructor, the exter-

Table 1

Corresponding student-centered CP functions from the WBS of the PIMIS project.

Who	Participative and collaborative functions (what)	Whom
Course instructor	CP1. Review the progress (P5)	Student team
	CP2. Guide project direction (P1)	Student team
	CP3. Resolve resources and conflict issues (P1)	Student team
	CP4. Report the progress (P5)	Instructor
Student team	CP5. Escalate conflicts such as resources and conflicts between the instructor and the company (P5)	Instructor, sponsoring company
	CP6. Acquire requirements (P1)	Instructor, sponsoring company
	CP7. Brainstorm and consolidate innovative ideas (P1)	Student team
	CP8. Assign tasks and obtain commitment (P2)	Student team
	CP9. Test and perform peer review (P3)	Student team
	CP10. Release product baselines (P4)	Student team
Company stakeholder	CP11. Confirm the requirements (P1)	Student team
	CP12. Answer ad hoc company specific questions (P1)	Student team
	CP13. Validate the implementation (P1)	Student team

nal stakeholder and the two (development and research) student sub teams. They centered on the student role to discuss the CP functions with other stakeholder roles. To further identify CP functions that are mandatory, the game followed a *lazy* strategy: each role player tried to be as lazy as possible and to persuade other role players to accept the proposed mandatory CP functions in which they should collaborate or participate. Table 1 shows the identified student-centered CP functions resulting from the game playing.

In order to model CP functions in meeting classes, the PIMIS project grouped these CP functions according to two rules: (1) same stakeholders for different CP functions and (2) the consecutive position of the CP functions on the WBS. Table 2 provides a list of the resulting meeting classes inside a development cycle based on the

CP functions grouping. The instructor then showed a LRC (Linear Responsibility Chart) to the project team to formalize stakeholder participation by assigning their degrees of participation in these meetings. Table 3 shows the assignment results.

In this project, meeting classes institutionalized the CP functions in two ways. Firstly, to be able to repeat the process, CP functions are encapsulated as the predefined agenda into a functional meeting class. In this way, when meetings are called, the designated members are aware of, and familiar with, the process. Secondly, a meeting is not considered to be complete until all of the agenda items have been accomplished. This means that when all of the tasks are not addressed in one meeting action (object); they are continued in subsequent meetings (instantiated meeting objects for the same meeting type) until all of the tasks are accomplished.

Table 2

Meeting classes in each development cycle.

Meeting classes	Functional descriptions (agenda)	CP
Review and requirement brainstorming (REQR)	The instructor reviews the implementation results for the previous development cycle. New requirements for the current cycle are developed and consolidated	CP1, CP2, CP3, CP4, CP5, CP6
Requirement planning and task assignment meeting (PP)	The team clarifies and discusses how to fulfill the requirements, and schedule the cycle time. Implementation tasks are assigned and commitments are obtained from each member	CP8
External requirement discussion (REQE)	The instructor confirms the requirements, discusses questions and clarifies the intended environment for the company	CP6, C11, CP12, CP13
Internal testing and validation meeting (VAL)	Team members test the implementation, and provide necessary help mutually prior to the instructor's review. They also discuss and finalize the ideas from the INNOV meetings prior to submission at the review meeting (REQR)	CP9, CP7
Configuration control meeting (CM)	The team collaboratively builds up a baseline of product content (including code, Web pages, databases, and documents) related to the development cycle. Peer review of current cycle documents (e.g., use cases, UCD, activity diagram for the use case, the revised cumulative ERD) is conducted to ensure the consistency between the document and system	CP10
Innovation brainstorming (INNOV)	The research subgroup manages innovative ideas for the project including brainstorming, as well as collecting and consolidating possible innovative features in this meeting	CP7

Table 3
Formalizing stakeholder participation in the meeting classes.

Meeting class	ESNE	Instructor	The student team					
			Sub group #1			Sub group #2		
			S1	S2	S3	S4	S5	
REQR		●	●	●	●	●	●	●
PP			●	●	●	●	●	●
REQE	●	○	●	⊙	⊙	●	●	⊙
VAL			●	●	●	●	●	●
CM			●	●	●	●	●	●
INNOV	○	○	⊙	○	○	●	●	●

Notes: (1) degrees of participation are illustrated as: ○: low involvement (to be notified of the meeting result), ⊙: medium involvement (optional to attend the meeting), and ●: high involvement (required to attend the meeting) and (2) in this case, S1, S2, S3, S4, and S5 denote the student members of the PIMIS project team.

3.3.2.3. Organize the meeting classes to form MF. In this approach, meeting classes were further interconnected into two types of MF. The first type is temporal meetings-flow and is formed according to the sequential relationships of the meeting classes on the WBS. The second type is contextual meetings-flow. It is defined according to the information/agenda dependences among the functional meeting classes. For example, running meeting class *A* requires some information which is concluded in another meeting class *B*. In this case, *A* and *B* have a contextual relationship, and during the execution of meeting *B*, members become aware that the agenda and corresponding conclusions will be used in some other meetings involving some other stakeholders. To construct the contextual flow, the PIMIS student team was taught to use the design structure matrix (DSM) (Mantel et al., 2008; Eppinger, 2001) to model the information dependences among the meeting classes. The correlation became the expected codified inputs and outputs of functional meeting classes, enabling a holistic understanding of the anticipated contribution of the meetings to the project as a whole. Fig. 2 shows the two flow results. The functional meeting class, as denoted by a cloud, presents the ad hoc or informal communications that center on the meeting to support and accomplish the given agenda.

There were various reasons for constructing these two flow types. When planned project meetings are based on the WBS, they naturally form a sequential relationship. A macro-level group process can then be identified once such a temporal meetings-flow is formed. The recognition of these group processes as well as the *when* (refer to the 5W1H method) of the meetings on the flow provides a guide to students in carrying out, and to the instructor for monitoring, designated collaborations through the project development. In addition to the sequential manner, the contextual meetings-flow helps to establish the role each functional meeting plays in the evolution of the project's group memory. Recognizing the *why* in planning functional meetings and understanding the context of various functional meetings help participants to stay focused on the agenda. This results in the smooth execution of meetings and the full participation of stakeholder throughout the project development.

3.3.3. Executing the MF

3.3.3.1. Execute the meetings on the flow. Using the cyclic development and “divide-and-conquer” strategy, each development cycle started with the project team being given some tasks in the REQR functional meeting. The instructor also reviewed the results of the previous cycle in this meeting. In REQR, because of meetings' contextual flow (i.e., the interdependence among meeting classes), students understood that this meeting should generate an agreed-to-do function list for the two follow-up meetings: (1) REQE meeting for them to meet external stakeholders and present ideas which should have been discussed with the instructor and (2) PP meeting to plan the implementation accordingly.

During the REQE meeting, the external stakeholders might express different opinions regarding the action items (to-do list) which the students and instructor had agreed upon. In this case the students checked to see if the to-do-list should be modified and then notified (via an informal communication, e.g., MSN or a phone call) the instructor. Although the instructor might choose to contact the stakeholder for further clarification or to attend the next round of REQE meetings, students were encouraged to work directly with the external stakeholders to resolve any disagreements. This would ensure that they gained valuable, real-world communication experiences.

After REQR and REQE, the student team called for the PP meeting to plan the to-do list, calculate time needed for this loop (ranging from 1 to 2 weeks depending on complexity) and to schedule VAL and CM meetings. In the PIMIS project, meetings were held many times during a development cycle. For example, the INNOV meeting was often held several times during a cycle to in order to continually brainstorm and merge ideas. Besides predetermined items, dynamic issues were also added to the agenda when necessary, so that the group handling the issues could have timely notification.

3.3.3.2. Operate a meeting. In addition to some basic meeting instructions, the instructor shared a meeting guide (O'Rourke, 2008) with students. The field guide provides a vivid description of several prototypes of people commonly encountered in meeting situations. As well as this educational role-playing, the instructor also highlighted the need for flexibility in conducting a meeting. Lastly, as well as discussing the traditional same-time same-place meeting mode, the instructor also encouraged students to seek out their own methods of running meetings more efficiently and conveniently, as well as to maintain flexibility.

3.3.3.3. Manage meeting dynamics. Software development is highly dynamic; therefore, during the development process, some unexpected occurrences may affect the conclusions reached in a previous meeting. For example, the external stakeholders may request changes at any stage during a development cycle with a resulting impact on the previously concluded agenda of some prior meetings. The MF approach assisted students in handling meeting dynamics using Fig. 2 and Table 3. The PIMIS team ran a change request meeting (a.k.a. a CCB meeting) to discuss the change, and used the contextual-flow in Fig. 2 to analyze the impact on any previously concluded meetings according to the information dependences of the contextual flow. The PIMIS project then used Table 3 to notify the members (according to Table 3) of these affected meetings so that they could check whether the previously concluded agenda needed to be revised (i.e., to make a minor update) or reworked (i.e., to reconvene the meeting). By doing so the project was able to justify and sustain the group effort on the reworking of meetings for a substantial change, or to keep members updated and informed regarding a minor change. Frequently, for

tasks/events that require CP function, the group effort is gradually skipped or is later handled by individuals during times of change in project development. The MF approach not only resolves this by institutionalizing group practice for the tasks/events that require teamwork, but also helps to sustain such group effort in managing changes under the dynamic development environment.

4. Quantitative investigation

4.1. Overall results

The PIMIS student team used the MF approach and completed its project with the successful implementation and delivery of the required system. Project requirements were completed 5 weeks ahead of schedule, thus allowing the team to work on additional functions. Table 4 shows the recorded efforts for running the meetings during 21 cycles over the six-month period of system development. Fig. 3 illustrates collective views regarding the collaborative effort by students, instructor, and company stakeholders in meetings during the project development period. Because the last three cycles were mainly for documenting the reports and the final presentation and not for the development of the system, they were removed in Fig. 3.

Table 4 and the left-hand diagram in Fig. 3 indicate that the instructor's effort gradually decreased as the project progressed. These results show the feasibility of the MF for a university professor with limited teaching capacity time. The effort manifested by the student team, however, did not diminish as they became more familiar with the development environment. Although this finding conforms to an increased effort in an incrementally built software development model, another reason was that students tended to code together during meetings. Therefore, the time for VAL and CM increased as the system grew, resulting in the total increase in team meeting effort.

4.2. Investigations of the approach

In addition to the overall result described above, we further looked into the development and investigated the effects of the approach on the development of this specific project case. In this study, the in-process benefits to be evaluated were taken from two perspectives: product quality effectiveness and project process efficiency.

In-process product quality effectiveness is defined in this study as the effectiveness in revealing and removing defects by the meetings-flow during the development. With regard to pro-

cess efficiency, these meetings were established according to the sequential and contextual relationships. We wanted to establish whether such meeting connectivity helps to control the in-process progress of each development cycle by looking at the 18 development cycles. Thus in this study, the efficiency of each cycle's progress is attributed to the stabilization of in-process control, which is enabled by the increased granularity of team progress, as illustrated by the meetings' statuses on the flow. Within these contexts, the investigation can be expressed by the following two research questions:

Research question I: To test whether the approach significantly helped to control the product quality within the project as the development scale increased.

Research question II: To test whether the flow of meetings significantly helped the in-process team progress control during each development cycle of the project.

To discuss these two questions, we used statistical analysis and hypothesis testing. Data for the analysis are presented in Table 5. The data items used in this analysis are defined as follows (note that the defects include both code defects and design defects in the documents):

- S_i : development scale for cycle i ;
- S_{it} : accumulated development scale (including both code and documentation);
- D_i : number of total defects revealed and removed by the meeting series in cycle (i);
- D_{ci} : number of defects resulting from current cycle (i) is revealed and removed in cycle i ;
- SPI_{ri} : the value of the schedule performance index (SPI) for cycle i ; it is observed in the REQR review meeting;
- SPI_{ppi} , SPI_{vali} , SPI_{cmi} : the SPI values recorded in the PP, VAL and CM, respectively, meetings of cycle i .

The measurement of the aforementioned data items is further noted as follows. The development scale (S_i) of each iterative development can be obtained by first counting the lines of code implemented for the development cycle. Unlike the real-world professionals who are able to produce more concise codes, students may write many lines of program code for a simple programming task. Hence, in addition to program size (line of code), students were taught the complexity factor in comprehensively assessing S_i . DeMarco's taxonomy of program complexity or the 14 evaluation questions from the function point method (DeMarco, 1982;

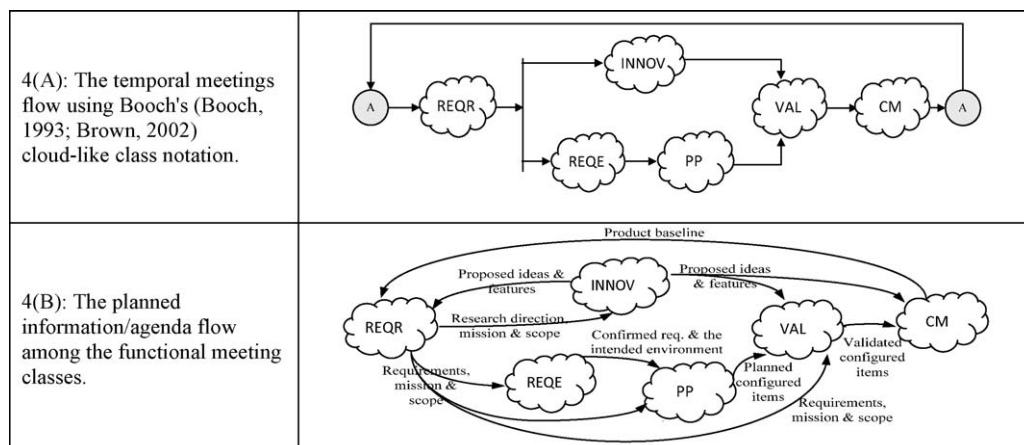


Fig. 2. The MF group process model for the PIMIS project case.

Table 4
Meetings effort for the 21 development loops.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
REQR	2	1.8	1.8	1.5	1.6	1.2	1.25	1.2	1	0.8	0.8	0.75	0.8	0.9	0.75	0.6	0.6	0.5	0.8	1.6	1.5
REQE	2	2.5	1	2.25	1.2	2.1	0.75	2	1.75	2	1.8	2.5	2.75	2.5	1.75	2	2.5	2	3.2	2.5	1.75
PP	1.25	1	1.5	1.5	1.2	1.25	0.8	1	1.25	1.2	1	0.8	0.75	0.75	0.5	0.6	1.5	1	0.8	1.2	1.5
CM	2	2.25	2	0.5	2	2.5	1.75	1.25	1.5	1.25	2	2.25	5	2.5	2	2	5.5	5.25	1.2	1.5	0.8
VAL	1.2	2.25	2.5	3	2.75	3	3.25	4	3.5	4.25	4	4.2	5	5	6	5.5	5.5	4	1.8	1.25	2.6
INNOV	2	2.25	2	2.5	2.75	2.5	3.25	3	3.75	4	4.5	3.75	4.25	4	4.5	4.5	2.25	1.5	0	0	0
Measurement unit: h; ■ : merging of meetings; E_{mi} and E_{ii} : accumulated effort on MF for the student team and the instructor																					
E_{mi}	10.45	9.8	10.8	11.25	11.5	12.55	11.05	12.45	12.75	13.5	14.1	14.25	13.55	15.5	15.2	12.85	14.25	8.2	8.05	10.55	
E_{ii}	2	1.8	1.8	1.5	1.6	1.2	1.25	1.2	1	0.8	0.8	0.75	0.8	0.9	0.75	0.6	0.6	0.5	1.2	1.6	1.5

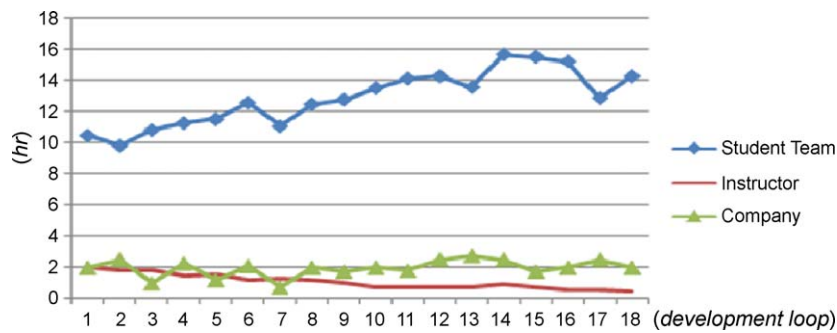


Fig. 3. Monitoring relevant stakeholder involvement in meetings.

Pressman, 2004; van Vliet, 2000) is good material for students to exercise complexity evaluation. For simplification, the evaluation scores obtained from the questions were mapped into three complexity classes for easy, moderate and more challenging functions.

With the meetings-flow design, students and the instructor took advantage of the designed meetings and worked together to clarify, review, and test ad hoc in-process products (see the meetings' operational definition in Table 2). During such a collaborative QA process, possible bugs were identified and solved before they moved on to next meetings. To measure the defects, students recorded the number of defects (D_i) identified in each meeting during the development. They further justified whether the defects were produced in the current cycle (D_{ci}). The derived value ($D_i - D_{ci}$) represents the number of missed defects in the previous cycle. Students were also asked to record the data of schedule performance index (SPI_{ti}) for the entire development cycle. It was obtained in the project case by calculating the completion rate of the to-be-implemented requirements. The SPI_{ppi} , SPI_{vali} , and SPI_{cmi} are defined as the completion rate of the planned meeting agenda corresponding to the implementation of the requirements. For example, suppose 6 out of planned 8 agenda items of the validation meeting are accomplished; the SPI_{vali} for this meeting is $6/8 = 0.75$.

Data of program size, complexity, schedule performance index, and the defects for each development cycle were recorded in students' engineering log book. In the project case, during the REQR meetings, the instructor may review these data values as well as the

code content (to verify the program size) with students to establish the objectivity of the measured data. Students are expected to benefit from this educational review with resulting improvement of programming skills and the estimation capability.

To address the first question, we use the defect removal effectiveness measure. Defect removal effectiveness, or called defect removal efficiency by some researchers (Kan, 2002, p. 124; Jones, 2008), is a way that can be used in a software project to assess a team's ability to find errors before they are passed to the next framework activity (Pressman, 2004, p. 663). In this study, we considered the two-level defect removal effectiveness (denoted as E_i) at cycle $i = D_i / [(D_{i+1} - D_{ci+1}) + D_i]$ and the corresponding defect miss rate (or called the Type II error (error of omission) rate, denoted as E'_i) for failing to find defects that are produced in current cycle i : $1 - E_i = (D_{i+1} - D_{ci+1}) / [(D_{i+1} - D_{ci+1}) + D_i]$. In the PIMIS project, the team was given a set of functions to implement for each cycle. The development scale (S_i) for cycle i is defined as $\sum_j (L_{ij} R_{ij})$, where the L_{ij} is the size (in KLOC) for a function set (j) and R_{ij} is the complexity level of the function set j . In this case, the complexity level was defined as 0.8, 1, and 1.2, respectively, for easy, moderate and more challenging functions. Note that the values of the complexity levels may vary due to different project settings and experiences. Therefore, the accumulated development scale at cycle i can be expressed as: $\sum_i \sum_j (L_{ij} R_{ij})$, and is further denoted as S_{it} . Normally, the number of produced defects increases as S_{it} grows and gets more complex (Jones, 2008; Jiang et al., 2007). Without attention, such an increase often results in an increase in E'_i . It is indicated that the defects,

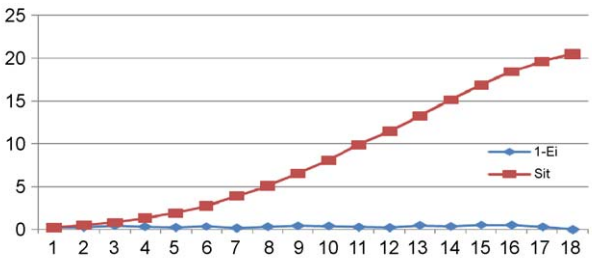
Table 5
The numerical report for the selected 18 development loops.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
S_i	0.2	0.3	0.32	0.5	0.6	0.8	1.2	1.15	1.5	1.52	1.8	1.6	1.75	1.88	1.7	1.56	1.2	0.88
S_{it}	0.2	0.5	0.82	1.32	1.92	2.72	3.92	5.08	6.54	8.06	9.89	11.44	13.24	15.12	16.82	18.38	19.58	20.46
D_i	8	11	12	16	15	11	14	10	12	14	15	10	6	7	6	8	9	5
D_{ci}	0	9	8	6	7	6	8	7	7	5	5	4	3	1	2	1	0	1
SPI_{ppi}	0.85	0.72	0.88	0.78	0.9	0.85	0.95	0.75	0.95	0.85	0.9	0.94	0.8	0.86	0.85	0.9	0.92	0.95
SPI_{vali}	0.9	0.92	0.85	0.82	0.92	0.9	0.95	0.96	1.02	0.95	0.9	1	0.95	0.93	0.9	0.85	0.96	0.99
SPI_{cmi}	0.96	0.95	0.9	0.87	0.9	0.95	1	0.96	1.05	0.93	0.93	1	0.98	0.95	0.95	0.97	0.95	1
SPI_{ti}	0.96	0.95	0.92	0.96	1	0.97	1	0.99	1.1	1.08	0.98	1.05	1.06	1	0.98	1	0.99	1.05

Table 6

The correlation analysis.

Regression test	
R	0.098776
R square	0.009756
Standard error	7.320343
Observations	18
Coefficient	
Intercept	0.315145
M	0.001894



identified through the meeting series, are controlled as the project evolves and the product scale increases. Hence the stability of the defect removal effectiveness due to the MF, is examined by comparing the value of E'_i and S_{it} to see whether E'_i can be significantly controlled as S_{it} increases.

According to the above statements, we used correlation techniques to analyze the correlation between the two variables, E'_i and S_{it} . Correlation was used to test the statistical significance of the association. Correlation coefficient, R , is a measure of the linear relationship that exists between the values of two variables. The closer the value is to +1, the stronger the positive correlation is between the two variables. A value close to zero indicates that there is a random, nonlinear relationship between the error of omission rate and the development scale. After computation (Table 6), R is 0.098776, which indicates that there is minimal linear correlation between the two variables. There is no positive correlation between E'_i and S_{it} . It is concluded that to the PIMIS project, the MF significantly helped to control the product quality as the scale of the product increased.

To address the second question, although the PIMIS project was successfully completed, we focused on the shift of schedule performance values indicated by the serialized meetings on the flow, and tested whether they formed a significantly positive influence on the final progress of each development cycle. In other words, we examined whether the organized flow of meetings generated a positive influence on controlling the in-process progress of each of the 18 cycles.

In project management, the schedule performance index (SPI) is often used to measure the progress performance of tasks (Marchewka, 2009). According to the definition, SPI is measured by the expression: *earned value/planned value*. For each development cycle, the REQR meeting serves as the final review and acceptance test. The value of SPI_{ri} indicates the final progress check for task completion of the entire cycle i . In addition, PP, VAL and CM meetings are checked for progress status. They are denoted as SPI_{ppi} , SPI_{vali} and SPI_{cmi} , respectively. Suppose the series $\{SPI_{ppi}, SPI_{vali}, SPI_{cmi}, \text{ and } SPI_{ri}\}$ is denoted as $\gamma(i)$, the hypothesis for the second question is then formulated as follows:

H₀. The sequence of numbers in $\gamma(i)$ has an ascending order.

H₁. The sequence of numbers in $\gamma(i)$ lacks an ascending order, or has a descending order.

To compute this analysis, we applied the Friedman test. The Friedman test is used to examine whether a consistent relationship exists among the tested data sets of ranking. It is also applied to indicate the ascending or descending nature of the relationship. Applying the formula of the Friedman test, we first examined whether the 18 ranking value sets for the variable series, $\{SPI_{ppi}, SPI_{vali}, SPI_{cmi}, \text{ and } SPI_{ri}\}$, had a significant difference. Two derived hypotheses can be further expressed:

Table 7

The ranking of variables and the statistics for Friedman test.

Variable	Mean rank		
SPI_{ppi}	1.19	N	18
SPI_{vali}	2.06	Chi-square	45.436
SPI_{cmi}	2.83	df	3
SPI_{ri}	3.92	Asymp. sig	0.000
		Monte Carlo: low limit (95%)	0.153

H₂. The variable series has no significant difference.

H₃. The series has significant difference.

We computed the chi-square value 45.432, and the value was larger than $\chi^2_{0.975}(3) = 9.348$; thus the result falls into the rejection area (the right tail 0.025). Therefore, according to the results of the analysis (Table 7), we reject **H₂** and further determine that the 18 data sets show a significant difference with an ascending order in the series SPI_{ppi} , SPI_{vali} , SPI_{cmi} and SPI_{ri} . It is concluded that the organized flow of meetings generated a positive influence on controlling the in-process progress.

5. Discussion

Besides the quantitative investigation, we conducted focus-group interviews to discuss the qualitative findings regarding the effect of the approach on the students and stakeholders. The focus group interviews were based on semi-structured question formats that centered on how the MF aids in resolving the three issues described in Section 1. The interviews contained two discussion sessions; one involved students only and the other included the students, the instructor and the company. Each session had a facilitator who administered questions and a moderator who controlled the discussion environment (Kontio et al., 2004; Morgan, 1996). Findings and the lessons-learned from the discussions are summarized as follows.

5.1. Connect people together in a centralized group path

In the discussion regarding effectively and continuously monitoring teamwork, participants asserted that, because the meetings and meetings-flow were clearly defined, people involvement could be formalized and individual progress publicly scrutinized. With meetings-flow, project risks could be monitored on an ongoing basis. Furthermore, because meetings were defined inside the development “black box” (Tiwana, 2004), the status of an individual’s commitment was more obvious, which helped to reduce the student “free-rider” and “Wyatt Earp” syndromes. By viewing meetings-flow as a centralized path of project teamwork, ad hoc and informal activities were brought up from such a backbone-like macro group process to support the collaborative development of the project. As one student responded, the existence of the flow

helped to structuralize teamwork proceedings; such a group path had the effects of connecting individuals, encouraging more assistance and accomplishing sharing.

With the establishment of such a group path, we conducted further surveys on the students and found that all showed a high degree of willingness to work as a team again. They felt that the MF held them together in the process of viewing, studying and resolving problems together.

5.2. Create a positive Hawthorne effect

In a structured batch, questions that asked how students felt when running a meeting and how the meetings-flow helped to encourage teamwork, one response reflected those of the others: “the MF brought the right information to the right people for a timely feedback and support”. In software development, a typical situation is that, before a problem gets out of control, developers are more likely to work alone rather than communicate and work with others (Heller, 2000). Although by nature, people like to help others who are in urgent need (Hudson, 2007), projects often fail because of the inability of people to know others’ need for help, which precludes giving appropriate help or support in a timely manner (Ceschi et al., 2005).

The MF raised this issue in the PIMIS case. During the discussion, the researcher also noted that the project team had encountered some members who were less enthusiastic about group work. To encourage participation and information-sharing in a meeting, they were taught to take advantage of elements in Maslow’s “Hierarchy of needs” model (Jones, 2007). Members were encouraged to share any success or interesting stories that might be personal or relevant to the team. Such encouragement helped them to obtain recognition and realize self-actualization. The instructor and the team also appreciated any new issues as valuable team assets; members were encouraged to speak out encountered problems. In addition, the value of helping people was emphasized and promoted with a team slogan: “How many times did you lend a hand today?” Students were encouraged to express appreciation to others, thus creating a positive Hawthorne effect. Under such conditions, individualized help became available and the team was able to promptly discuss any concerns before they escalated into roadblocks.

5.3. Link meeting performance to the grading policy

During the interviews, the students expressed a concern about linking meeting performance to the grading policy. To sustain the positive team environment, we found that students may desire incentives and additional feedback from the instructor. The instructor replied that he might have had a grading policy whereby extra credit could be earned according to the number of verbal opinions a student expressed in a meeting, as well as the number of times of helping others. The instructor provided such points as a form of encouragement, as opposed to punishment for incomplete work. This grading policy can be used where some skillful students may tend to act as “lone warriors” (Motschnig-Pitrik and Figl, 2007) and individualist (Smith et al., 2005). The extra-credit grading design does not punish these students but rather entices them to earn higher grades by helping others.

5.4. Streamline project meetings and the discussions

According to the company, the student team was very organized each time the REQE meeting was held. Sessions were fluid and well connected to prior discussions. Students were very consistent in their knowledge and comments during the discussion. The external stakeholder noted: “apparently they had gone through the issues together and had reached consensus before they came”.

The replies of the students towards the external stakeholder’s feedback were twofold. Firstly, “. . . during every REQR the instructor helped direct our thoughts before we presented them to company stakeholders in REQE, . . . and then we were supposed to bring the feedback from the discussion with company stakeholders to the next REQR”. By the flow of meetings the instructor was able to intervene in the external collaboration and watch how students interacted with the Company, thus helping to keep things on track. Secondly, the students felt that this was also because “for each meeting on the contextual flow, they were required to maintain the content and conclusion; these meetings’ documentations, once collected, presented the evolution of the project content agreed on by relevant stakeholders”. Through meetings-flow, students learned to build the group memory in an organized way and then communicate with the company stakeholders and the instructor in a consistent way, with proof and evidence. Project contents preserved in such a way are helpful when it is necessary to roll back to a certain version; students are able to check the relevant meeting contents to help recall the group memory.

During this discussion, students further suggested a computerized tool support for the MF. Because the PIMIS project did not have any CASE tool in helping to manage the project contents, students expressed a concern: although they did not encounter the problem, managing the documents of group work might become an issue for a more complicated project. Perhaps a computerized tool support for the MF could expedite the management of documentation work.

5.5. Related work

In addition to the discussion on using the meetings-flow approach in the project case, we further discuss the distinction between the current study and the earlier work of Chen (2009). We state the differences, as well as summarize the added values this paper has offered to the development of the approach.

Firstly, this paper extends earlier work by adding the contextual meetings-flow content to comprehensively present the meetings-flow approach. Secondly, this paper adds the in-depth technical content regarding the operational framework of the approach. This includes the detailed methods observed from the case study for identifying senior projects’ CP functions, defining the student-centered functional meeting classes, and organizing the flow of the meeting classes. Thirdly, the project case presented in this paper differs from the example and the data provided in the earlier work. Fourthly, this paper adds a formal empirical study to further investigate the use of the approach. Finally, the current study provides more comprehensive learned lessons from the aspect of software engineering education.

The project case in this paper focused on meetings and with a special treatment on project meetings to address the senior project-related issues raised in this study. In practice, there are several development methods, though they are not solely designed for student team projects, which are similar in nature. For example, the XP programming and SCRUM approach of the agile development methods promote test-first and continuous integration in order to meet customers’ needs efficiently (Beck and Andres, 2004). These features are similar to the short-iteration and incremental-implementation design of the meetings-flow approach. Besides, the XP and SCRUM also emphasize continuously communicating with the users or clients in meetings (Rising and Janoff, 2000); yet the meetings-flow approach further focuses on the interconnectivity of functional meetings.

Moreover, in order to effectively meet customers’ needs, the methods focus more on coding and face-to-face conversation, rather than documentation (Ruping, 2003; Rising and Janoff, 2000). Yet several other studies express different opinions affirm the necessity of project documents (Sillitti et al., 2005; Petersen and Wohlin,

2009; Selic, 2009). Despite the dispute as to whether documentation contributes to project success, the meetings-flow approach suggests the usability of the contextualized documentation of meeting minutes (due to the meetings contextual flow) as the footprints of the collaborative senior project development. As mentioned earlier in this section, the establishment of such an interconnection forms the flow of group memory, presenting how the teamwork proceeds and helping to streamline students' interactions with external stakeholders.

5.6. The connection between MF and an increase in quality

In this paper, we report on how a distinguished senior project case succeeds, using a special approach. This paper explores the in-depth technical content of the approach. This paper investigates the case and the approach by looking into the development of the project case to analyze how MF takes effect during the process and contributes to the project result. Inside the investigation, this paper analyzes the connection, in terms of group progress control and defect removal effectiveness, between the group success and the approach. In addition to current in-depth single case study with a project management perspective, the generalization of the benefit and other quality connections, such as the teamwork quality, may be another valuable research concern in the continual evolution of the approach. Therefore, in the next research action, we will conduct an extended experiment in breadth over a number of teams and a comparative study with other management approaches to discuss the benefits, especially the teamwork quality, generalized by the approach.

6. Validity check

In this section, the validity of the case result and the use of the focus group method are further addressed. The four perspectives: construct, external, conclusion, and internal validity issues (Kitchenham et al., 1995) are used in validating a single project case. Hence we take these perspectives to cope with the threats to the project case in this study.

To minimize the threats to the construct validity of the project case, the composition and complexity of the project case were reviewed. The PIMIS case had a comprehensive stakeholder composition of a senior project. In addition, the project team was moderately student-team sized but faced significant development time for the completion of the project. The complexity can also be seen from the industrial involvement perspective, in which students were required to develop a system to meet a company's IT needs. Moreover, in the case study, the senior projects at the university are the most important aspect of the curriculum for senior students. Hence the students can spend much of the time on senior project development. However, in a case when senior projects are not a major course, students' time devoted to senior projects may be less than that in the studied case. In this situation, the result might not be as good as that of the studied case.

To minimize threats to external validity, the study was conducted in the engineering and computing education domain, thus constituting a limitation in regard to application of the case. Furthermore, training is another critical issue related to external validity. Since the students in the PIMIS project case were inexperienced in project development and meeting operation, they were prepped at the beginning of the project. For similar situations, we suggest that training in project planning and meeting operation be given to ensure that individuals have sufficient knowledge for effectively executing the project and MF. The issue of institutionalizing the performance of the meetings-flow when considering applying the approach to other senior projects should be of concern.

We suggest that the repeatability of conducting the meetings-flow be further addressed using the CMMI's four common features (i.e., ability to perform, commitment to perform, and verifying and directing the implementation) and the associated generic practices (Bamberger, 1997; SEI, 2006) in order to institutionalize the practices established in the reported case.

The study's conclusion validity is addressed from two perspectives: the representativeness of the project case and data used in the case. For the first perspective, senior projects are a major focus of the software engineering education in the organization, and the case presented in this study has a typical configuration of the senior projects in the organization. As for the second perspective, results were comprehensively collected and confirmed with different methods from several different sources. For the data used in the research questions and the contents of the interviews, member checking (Seaman, 1999) was performed by allowing subjects to review the data and comment on transcripts of their interviews.

In this study, the investigation shows that the meetings-flow approach helped to remove in-process defects. Notable internal factors for such a conclusion, including whether other help existed in contributing to the result and the competence of the student team, should be addressed. The project case in this study did not use other commercial defect removal tools. Although students used some built-in debugging function in the development package for checking individual programming content, they relied on the designed meetings as the collaborative defective removal process to assure the quality of the built and integrated product contents. In addition, one presumable fact for contributing to the result might be that the students were very able. The student team members in this study were not pre-selected to form an exceptional team. They presented an average capability level of software programming prior to entering the development. The aforementioned internal confounding factors should be considered when applying this approach elsewhere.

Another issue that merits mentioning is that, in the in-process defect removal effectiveness investigation, the fraction of defects that is missed is determined primarily by the quality of the process being used to detect the defects. The investigation has shown that the meetings-flow process helps control the product quality. However, only the existence of the process does not stand for such a conclusion. To our knowledge, when the project and the product get more complex, the process should be further tailorable in order to better handle (i.e., maintaining the quality of the process) the complexity and dynamics. In this study, the tailoring capability and flexibility of process was seen inside the meetings-flow. For instance, during the initial stage of the project when the software product was not complicated, the team planned (in the PP meeting) to be more flexible in running the meetings, and their efforts spent on the meetings were low. As the product complexity increased, although the meetings-flow remained the same, more formal operation of the meetings was planned, and students did spend more efforts in the meetings (refer to Fig. 3). To us, the meetings-flow (MF) is a process framework, and students need to adjust and plan/tailor the content and operation of the MF during each development cycle. Such a process framework, besides serving as a group process to guide the collaborative development, should also facilitate the tailoring capability for really helping control the quality of the product.

In addition to the aforementioned internal validity issue, this study focuses on the threats from the use of the focus group method. According to Kontio et al. (2004), the threats of focus group method in software engineering studies include: group dynamics, social acceptability, hidden agenda, secrecy, and limited comprehension. To minimize the group dynamics threat, i.e., uncontrollable flow of discussion without predefined format of the agenda, this study conducted structured interviews on two groups: one was for the

student team only and the other for all of the stakeholders. The moderator also balanced the discussion and engaged passive participants. In group discussions, social acceptability to particular participants (e.g., the instructor) may affect the validity of the content. To mitigate this, the study took the nominal group technique (NGT) by making written feedbacks anonymous during discussions and having the moderator facilitate the balancing of oral comments.

To avoid hidden agenda and disclosure challenges, i.e., concealment of information due to conflict of interest, the interviews were held after the course was completed and final grades were posted. As for the limited comprehension, it refers to the insufficiency of the collected results due to interview time constraint, unfamiliarity within the participant group, and participants' comprehension ability. Morgan (1996) showed similar concerns in educational studies and suggested that focus groups are often conducted with existing groups for a more interactive discussion during the interview. In this study, the focus group was the PIMIS student team and external stakeholders who had been working together through eighteen weeks of project development. The researchers also prepped the participants to ensure that they would be able to understand the discussion.

7. Conclusion

This paper has presented a case study and reported a special approach used in the project case of conducting project-based educational collaboration in SE education. To resolve the three issues stated in this paper, the approach took the socio-technical aspect of software development by redefining meetings to form meetings-flow (MF). MF provides the case project with a continual educational collaborative environment. Numerical data and the focus group results indicated that both quantitative and qualitative values support the MF approach in effectively conducting a senior project like the PIMIS case.

This paper reports on a distinguished project case and has an exploratory nature with an in-depth introduction to the meetings-flow approach used in the project. In the follow-up research, we would like to extend the study by conducting a number of student projects to investigate the more generic benefits of the approach. This includes the experimental comparisons with the projects where the MF approach was not used. Besides, we would like to explore the possibility of developing an information system to support the overall execution of the meeting-oriented senior project management tasks.

Acknowledgements

The authors express their appreciation to Mr. H.A. Liu for his research assistance and contribution in the early preparation of the paper; Mr. K.H. Chao and the PIMIS project team for providing the data; and ESNE Inc. for offering necessary resources in conducting the project. Finally, the authors thank the Taiwan National Science Council for financially supporting this study (95-2221-E-182-011). Finally, we also thank the anonymous reviewers for their contributive comments.

References

Abernethy, K., Piegari, G., Reichgelt, H., 2007. Teaching project management: an experiential approach. *Journal of Computing Science in Colleges* 22 (3), 198–205.
 Bamberger, J., 1997. Essence of the Capability Maturity Model. *IEEE Computer* 30 (6), 112–114.
 Beck, K., Andres, C., 2004. *Extreme Programming Explained: Embrace Change*, 2nd edition. Addison-Wesley, Boston, MA, USA.
 Beck, K., Ward, C., 1989. A laboratory for teaching object oriented thinking. *ACM SIGPLAN Notices* 24 (10), 1–6.
 Boehm, B.W., Port, D., 2001. Educating software engineering students to manage risk. In: *Proceedings of the 23rd International Conference on Software Engineering*.

Brown, D.W., 2002. *Introduction to Object Oriented Analysis: Objects and UML in Plain English*, 2nd edition. John Wiley & Sons, Inc., New Jersey, USA.
 Ceschi, M., Sillitti, A., Succi, G., Panfili, S., 2005. Project management in plan-based and agile companies. *IEEE Software* 22 (May/June (3)), 21–27.
 Chamillard, A.T., Braun, K.A., 2002. The software engineering capstone: structure and tradeoff. *ACM SIGCSE* 34 (1), 227–231.
 Chen, C.Y., 2009. A meetings-flow approach for conducting student final-year projects. *Journal of Computing Science in Colleges* 24 (6), 28–34.
 Darke, P., Shanks, G., Broadbent, M., 1998. Successfully completing case study research: combining rigor, relevance and pragmatism. *Information Systems Journal* 8 (4), 273–289.
 Dawson, R., 2000. Twenty dirty tricks to train software engineers. In: *Proceedings of the 22nd International Conference on Software Engineering*.
 DeMarco, T., 1982. *Controlling Software Projects – Management Measurement & Estimation*. Yourdon Press Computing Series, Englewood, NJ, USA.
 Denning, P., 1992. Educating a new engineer. *Communications of the ACM* 35 (12), 83–97.
 Eppinger, S., 2001. Innovation at the speed of information. *Harvard Business Review* 79 (1), 149–158.
 Favela, J., Peña-Mora, F., 2001. An experience in collaborative software engineering education. *IEEE Software* 18 (2), 47–53.
 Fornaro, R.J., Heil, M.R., Tharp, A.L., 2007. Reflections on 10 years of sponsored senior design projects: students win-clients win. *Journal of Systems and Software* 80 (8), 1209–1216.
 Gallivan, M.J., Keil, M., 2003. The user-developer communication process: a critical case study. *Information Systems Journal* 13, 37–68.
 Goldratt, E.M., 1997. *Critical Chain*. The North River Press, Great Barrington, MA, USA.
 Hass, M.R., 2006. Knowledge gathering, team capabilities, and project performance in challenging work environment. *Management Science* 52 (8), 1170–1184.
 Hassan, A., 2008. A methodology for combining development and research in teaching undergraduate software engineering. *International Journal of Engineering Education* 24 (3), 567–580.
 Heller, T., 2000. If only we'd known sooner: developing knowledge of organizational changes earlier in the product development process. *IEEE Transactions on Engineering Management* 47 (3), 335–344.
 Hilburn, T.B., Humphrey, W.S., 2002. Teaching teamwork. *IEEE Software* 19 (5), 72–77.
 Hudson, V.F., 2007. The human touch. *Industrial Engineer* 39 (9), 40–44.
 Jiang, Z., Naudé, P., Jiang, B., 2007. The effects of software size on development effort and software quality. *World Academy of Science, Engineering and Technology* 34, 31–35.
 Jones, G.R., 2007. *Introduction to Business*. McGraw-Hill Inc., New York, NY, USA.
 Jones, C., 2008. Measuring Defect Potentials and Defect Removal Efficiency, Technical Paper of Software Productivity Research, LLC.
 Kan, S.H., 2002. *Metrics and Models in Software Quality Engineering*. Addison-Wesley, Boston, MA, USA.
 Kitchenham, B., Pickard, L., Pflieger, S.L., 1995. Case studies for method and tool evaluation. *IEEE Software* 12 (4), 52–62.
 Kontio, J., Lehtola, L., Bragge, J., 2004. Using the focus group method in software engineering: obtaining practitioner and user experiences. In: *Proceedings of the 2004 International Symposium on Empirical Software Engineering*.
 Kornecki, A.J., Hirmanpour, I., Towhidnadjad, M., Boyd, R., Ghiorzi, T., Margolis, L., 1997. Strengthening software engineering education through academic industry collaboration. In: *Proceedings of the 10th Conference on Software Engineering Education and Training*.
 Lipman, M., 2003. *Thinking in Education*. Cambridge University Press, Cambridge, UK.
 Mantel, S.J., Meredith, J.R., Shafer, S.M., Sutton, M.M., 2008. *Project Management in Practice*. John Wiley & Sons, Inc., New Jersey, USA.
 Marchewka, J.T., 2009. *Information Technology Project Management*. John Wiley & Sons, Inc., New Jersey, USA.
 Mead, N.R., 2009. Software engineering education: how far we've come and how far we have to go. *Journal of Systems and Software* 82 (4), 571–575.
 Mead, N.R., Beckman, K., Lawrence, J., O'Mary, G., Cynthia Parish, C., Unpingco, P., Walker, H., 1999. Industry/university collaborations: different perspectives heighten mutual opportunities. *Journal of Systems and Software* 49, 155–162.
 Melin, U., Cronholm, S., 2004. Project oriented student work – learning & examination. In: *Proceedings of the 9th SIGCSE Conference on Innovation and Technology in Computer Science Education*.
 Moore, D., Berry, F., 1999. Industrial sponsored design projects addressed by student design teams. In: *The 29th Annual Conference on Frontiers in Education*.
 Morgan, D.L., 1996. Focus groups. *Annual Review of Sociology* 22, 129–152.
 Motschnig-Pitrik, R., Figl, K., 2007. Developing team competence as part of a person centered learning course on communication and soft skills in project management. In: *Proceedings of the 37th ASEE/IEEE Frontiers in Education Conference*.
 O'Rourke, M., 2008. A field guide to meetings. *Risk Management* 55 (5), 61.
 Petersen, K., Wohlin, C., 2009. A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *The Journal of Systems and Software* 82, 1479–1490.
 Pressman, R., 2004. *Software Engineering Software Engineering: A Practitioner's Approach*. McGraw-Hill Inc., New York, NY, USA.
 Reichlmayr, T.J., 2006. Collaborating with industry – strategies for an undergraduate software engineering program. In: *Proceedings of the 2006 International Workshop on Summit on Software Engineering Education*.

- Rising, L., Janoff, N.S., 2000. The SCRUM software development process for small team. *IEEE Software* 17 (4), 26–32.
- Rooji, S.W., 2009. Scaffold project-based learning with the project management body of knowledge. *Computers & Education* 52 (1), 210–219.
- Ruping, A., 2003. *Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects*. John Wiley & Sons, Inc., New York, NY, USA.
- Sancho-Thomas, P., Fuentes-Fernandez, R., Fernandez-Manjon, B., 2009. Learning teamwork skills in university programming courses. *Computers & Education* 53, 517–531.
- Schlimmer, J.C., Fletcher, J.B., Hermens, L.A., 1994. Team-oriented software practicum. *IEEE Transactions on Education* 37 (2), 212–220.
- Scragg, G., Baldwin, D., Koomen, H., 1994. Computer science needs an insight-based curriculum. *ACM SIGCSE Bulletin* 26 (1), 150–154.
- Seaman, C., 1999. Qualitative methods in empirical software engineering. *IEEE Transactions on Software Engineering* 25 (4), 557–572.
- SEI, 2006. *Capability Maturity Model Integration*. Carnegie Mellon University Press, Pittsburgh, PA, USA.
- Selic, B., 2009. Agile documentation, anyone? *IEEE Software* 26 (6), 11–12.
- Sikkel, K., Spil, T.A.M., van de Weg, R.L.W., 1999. A real-world case study in information technology for undergraduate students. *Journal of Systems and Software* 49, 117–123.
- Sillitti, A., Ceschi, M., Russo, B., 2005. Managing uncertainty in requirements: a survey in documentation-driven and agile companies. *IEEE Intl. Conf. on Software Metrics*.
- Smith, K.A., Sheppard, S.D., Johnson, D.W., Johnson, R.T., 2005. Pedagogies of engagement: classroom-based practices. *Journal of Engineering Education* 94 (1), 87–101.
- The Standish Group, 2007. *CHAOS Summary for 2006*. West Yarmouth, MA, USA.
- Tiwana, A., 2004. Beyond the black box: knowledge overlaps in software outsourcing. *IEEE Software* 21 (5), 51–58.
- van der Duim, L., Andersson, J., Sinnema, M., 2007. Good practices for educational software engineering projects. In: *Proceedings of the 29th International Conference on Software Engineering*.
- van Vliet, H., 2000. *Software Engineering Principles and Practice*. John Wiley & Sons, Inc., New Jersey, USA.
- van Vliet, H., 2006. Reflections on software engineering education. *IEEE Software* 23 (3), 55–61.
- Vaughn, S., Schumm, J.S., Sinagub, J., 1996. *Focus Group Interviews in Education and Psychology*. Sage Publications, Thousand Oaks, CA, USA.
- Wenger, E., McDermott, R., Snyder, W., 2002. *A Guide to Managing Knowledge: Cultivating Communities of Practices*. Harvard Business School Press, Boston, MA, USA.
- Wohlin, C., orn Regnell, B., 1999. Strategies for industrial relevance in software engineering education. *Journal of Systems and Software* 49, 125–134.
- Yin, R., 2008. *Case Study Research: Design and Methods*. Sage Publications, Thousand Oaks, CA, USA.
- Yoshioka, T., Herman, G., Yates, J., Orlikowski, W.J., 2001. Genre taxonomy: a knowledge repository of communicative actions. *ACM Transactions on Information Systems* 19 (4), 431–456.
- Chung-Yang Chen** received his PhD degree from Dept. of IE, Arizona State University, USA. He is currently an assistant professor in the Department of Information Management (IM) at National Central University and is jointed appointed by the Department of IM at National Taiwan University, Taiwan. Dr. Chen is also serving as a director of Taiwan Electronic Business Management Society. His research interests include project management, software engineering & education, information quality, and operations research. He has papers published in *International Journal of Project Management*, *IIE Transactions*, *Journal of Systems and Software*, *Computers and Education*, *Expert System with Applications*, *International Journal of Electronic Business Management*, *International Journal of Operations Research*, etc.
- P. Pete Chong** is the professor jointed appointed by the Department of Industrial Engineering Management and Department of Industrial Design at Ming chi University of Technology, Taiwan. Prior to returning to Taiwan, he was the Martel Chair Professor in MIS at University of Houston-Downtown. He also taught at Gonzaga University, University of Idaho, and Southeastern Louisiana where he also headed the Business Research Unit. Dr. Chong's research interests are in the modularization and integration of business systems, learning and idealization content design, and the theory and application of Pareto Principle.