# Enhancing the Student Learning Experience in Software Engineering Project Courses

Maíra Marques, *Member, IEEE*, Sergio F. Ochoa, *Member, IEEE*,
María Cecilia Bastarrica, and Francisco J. Gutierrez

***Abstract*—Carrying out real-world software projects in their academic studies helps students to understand what they will face in industry, and to experience first-hand the challenges involved when working collaboratively. Most of the instructional strategies used to help students take advantage of these activities focus on supporting agile programming, which is appropriate for capstone courses. This is not always recommended in initial software engineering project courses, however, where novice developers run projects in teams while simultaneously taking other courses. To enhance the learning and teamwork experience in this latter instructional scenario, this paper proposes a formative monitoring method, reflexive weekly monitoring (RWM), for use in project courses that involve disciplined software processes and loosely coupled work. RWM uses self-reflection and collaborative learning practices to help students be aware of their individual and team performance. RWM was applied in a case study over nine consecutive semesters. The results obtained indicate that RWM was effective in enhancing the learning experience in the instructional scenario studied. While students in the monitored teams were more effective and coordinated, and experienced a higher sense of team belonging and satisfaction, little evidence was found of them being more productive than students working in non-monitored teams.**

***Index Terms*—Collaborative learning, disciplined software process, formative monitoring, project-based learning, reflexive monitoring, reflective practice, software engineering project course, team performance, teamwork.**

## I. Introduction

COMPUTER science programs strive to prepare future software engineers for work in industry, by teaching students core computing concepts that will allow them to become lifelong learners, able to keep pace with innovations in the discipline. Approaches to delivering technical knowledge have usually been well adopted by universities. However, the development of transversal capabilities, such as leadership, teamwork, decision-making, negotiation, and self-reflection, are usually less supported in these programs [1]–[3]. These team's capabilities, also known as "soft skills", not only impact the teams' results but also their work climate [4], since software development also involves several human and social aspects [5].

Universities are aware of this situation, and have been trying to create recipes to develop soft skills in future software engineers. There seems to be a consensus that project-based and capstone courses that imitate industrial processes can contribute to addressing this challenge, mainly because they show students the need for applying such skills [6]–[9].

Although project-based learning [10] is the most common instructional approach used to implement these experiences [11], [12], instructors must decide what software development strategy to use for this. Typically, the chosen strategy ranges between disciplined and agile processes [13], the former being structured and based on phases and roles, and the latter being unstructured, focused on the product, and based on self-organized teams. Both strategies have shown positive results [11], [12], [14], but have different purposes and requirements. Disciplined processes are useful for inexperienced developers [13], and are recommended in project-based courses that help students internalize their previously-acquired theoretical software engineering knowledge [15]. This development strategy supports the asynchronous and distributed work of students, who perform individual activities most of the time followed by sporadic coordination tasks. This loosely-coupled work style provides flexibility to students to contribute to a project, while they are still taking other courses.

Agile processes, on the other hand, seem to be more appropriate for supporting capstone courses [16], where students gain a broader perspective of the software development practice, and reinforce their technical and non-technical skills, before being recruited into the software industry. These processes require students to have ample time to work together, since their activities are more tightly-coupled than in disciplined developments. Moreover, these students should have some prior experience in developing software [15].

Several mechanisms have been reported in the literature to enhance students' experience when using agile methods. However, less research has been done on supporting the learning process when using disciplined development strategies. To advance knowledge in this instructional scenario, this paper proposes the *Reflexive Weekly Monitoring* (RWM) method, which enhances students' learning experience during project-based courses using disciplined software strategies. RWM assumes that students will perform loosely-coupled

Manuscript received January 22, 2016; revised October 3, 2016 and April 30, 2017; accepted July 31, 2017. Date of publication September 6, 2017; date of current version February 1, 2018. This work was supported by the Project Fondef IDeA under Grant IT13I20010. The work of M. Marques was supported by the Ph.D. Scholarship Program of Conicyt Chile (CONICYT-PCHA/Doctorado Nacional) under Grant 2012-21120544. *(Corresponding author: Maíra Marques.)*

The authors are with the Department of Computer Science, Universidad de Chile, Santiago 8370459, Chile (e-mail: mmarques@dcc.uchile.cl; sochoa@dcc.uchile.cl; cecilia@dcc.uchile.cl; frgutier@dcc.uchile.cl).

Digital Object Identifier 10.1109/TE.2017.2742989

work during the course of the project. This monitoring method uses self-reflection and collaborative learning practices to help students realize how well they are working from an individual and team perspective; this gives them the opportunity to learn from past experience and address the remaining work in a more appropriate way. After informally applying this monitoring method in an undergraduate software project course for two years, it was formalized in order to evaluate its real benefits in practice. The evaluation process considered the following hypotheses:

H1: RWM positively impacts coordination among team members,

H2: increases students' sense of belonging to a team,

H3: makes them work more effectively, and

H4: increases their productivity.

*Coordination* is defined as the capability of team members to work together to reach project goals, and the *sense of belonging to a team* as the team members' perception of sharing a common goal. *Effectiveness* is understood as the capability to focus on the mandatory requirements of the software project [17]. Finally, *productivity* is measured by the amount of useful software built during the project.

To study the validity of the hypotheses, the RWM method was used in nine consecutive semesters –from spring 2011 to spring 2015– to monitor 18 out of 32 software development teams, which were all similar in terms of background, structure, and size. In total, 205 computer science undergraduate students participated in the evaluation. The projects addressed were comparable in terms of size, complexity, and duration, and all teams followed the same software development process [18].

The results indicate that RWM helps student teams improve their coordination, sense of belonging to a team, and effectiveness, but not necessarily their productivity. Monitored teams tend to be more productive during the first half of the project, but this changes when they realize that they have the project under control. From that point on, students tend to be more speculative, probably adopting the so-called "apprentice attitude" [19], [20], which negatively impacts on the team productivity. This issue requires further research and is planned as part of the future work. The empirical results have also shown some unexpected but valuable results, such as the early detection of free riders and of project risks.

From a personal perspective, monitored students felt comfortable with the process and believed that it helped them better address the project. Moreover, these teams felt more satisfied with the completed work than those not monitored.

The rest of the paper is structured as follows. The next section discusses related work. Section III describes the educational context. Section IV presents the RWM method. Section V describes the empirical setting designed for studying the validity of the stated hypotheses. Sections VI and VII report and discuss the results, respectively. Section VIII presents the study threats to the study's validity. Finally, Section IX concludes the paper and provides perspectives on future work.

## II. Related Work

The literature reports several strategies for addressing project-based learning experiences in software engineering courses, such as simulations and the development of hypothetical and real software products [21]–[23]. Project-based learning is probably the most valuable strategy for students and instructors, but is considered time-consuming and difficult to implement [8], [24]. When these instructional activities are implemented, students should take advantage of them, although this can represent a challenge for instructors.

Several methods have been proposed to help students acquire technical and non-technical skills during these experiences. Some were conceived to be used with agile processes, others with disciplined processes, and others are independent of the development strategy. The position of the course within the curriculum, and the level of activity coupling required of students in these projects will encourage the adoption of a particular development process. In turn, this determines the instructional strategy and tools used to support students acquiring technical and non-technical skills throughout the project. Typically, the first project-based course in a curriculum tends to follow a disciplined strategy, with well-defined life cycle phases and milestones, templates for the artifacts required, and students' work being loosely-coupled [13].

To enhance project-based learning experiences in this context, some researchers propose the use of e-portfolios where students store their own work products, and keep a logbook with their activities and reflections on a particular project [25], [26]. These portfolios, which can be used with both disciplined and agile development strategies, are accessible to the instructor, who can provide feedback and assess the students' performance. Despite the positive results reported in [26], the use of e-portfolios can create tension between the privacy of the students' thoughts and their knowing that these are to be shared with the instructor [25], which may lead to discomfort for students. Moreover, when the e-portfolio information is used as part of the students' assessment, there is little collaboration between students because of plagiarism concerns [25], which further reduces the chances to reinforce learning during these practical experiences.

With the same goal, Mahniè and Èasar [27] propose the use of a tool, similar to an e-portfolio, that visualizes students' activities in Scrum-based software engineering capstone courses. The visual display of task accomplishment not only allows monitoring of students, but also prevents procrastination and the "free-rider" syndrome. This tool seems appropriate for monitoring students, but was conceived particularly for teams using an agile software development strategy; it is therefore probably unsuitable for disciplined developments, since both strategies have different requirements and different purposes in the academic scenario.

Coaching, in which an experienced developer or instructor assists student teams while they work on a project, is another common practice to improve learning experiences, and is suitable for both disciplined and in agile developments. Of the several coaching approaches, the most typical focus on helping students address internal issues (e.g., communication and coordination) by mentoring them to use their knowledge and

review their deliverables [24], [28]. Rodríguez *et al.* [29] propose an agile coach with pedagogical expertise, who monitors the students' accomplishments and suggests corrective actions and improvements. The evaluation results indicate that students who were coached following agile principles perceived a greater gain in non-technical skills than non-coached students. Despite the good results reported in the literature, this approach is more appropriate for capstone courses, since inexperienced students tend to rely on the coach, and avoid assuming the responsibility and risks of their projects. If the coach becomes a project manager, then students lose a good opportunity to learn more from these experiences.

Monitoring can also be used to make project experiences valuable for students and instructors. A monitor helps student teams to reflect, so as to address and solve their team and individual issues. Several researchers indicate that team monitoring reduces motivational losses and increases the likelihood of detecting free riding and social loafing [27], [30], and helps focus the team effort on reaching the team goals over individual interests [31]. Monitoring can operate as an implicit coordination mechanism [32], by making team members more aware of each other' actions, timing, and performance; students can thus increase their ability to synchronize their contributions to maximize the attainment of team goals [33]. This approach appears to be an effective way to promote product quality and team productivity [34].

Typically, in disciplined strategies, students conduct loosely-coupled work to address the project tasks. Due to the effort and complexity of monitoring their activities, software tools are used by instructors and teaching assistants. For instance, Collofello and Hart [35] proposed a tool where students report their project status weekly. However, this system relies only on information provided by the students, and therefore the diagnosis could be inaccurate or definitively wrong. A similar tool was proposed by Bakar *et al.* [36], which also depends on student input. This characteristic potentially undermines the usefulness of these systems, for several reasons. For instance, students tend to report their activities at the last possible minute, so report what they remember rather than what they have really done. In other cases, they report what is required to avoid conflicts with team members and course instructors.

More objective monitoring mechanisms have been proposed based on mining the log files recorded by the tools used by students for supporting their activities, such as, version control systems [37], [38], issue trackers [11], [39], wikis [39], or mailing lists [11], [39]. Although these mechanisms are useful, they only capture activities mediated by such tools; this could be appropriate for monitoring global software development scenarios, but is not completely transferable to project-based courses, where students and monitors frequently conduct face-to-face interactions that are not logged.

In the case of using Scrum as a software development strategy, Scrum Masters can perform the team monitoring. In particular, Mahnic [40] proposes that the instructor assumes that role in capstone courses by ensuring that everyone follows Scrum rules and practices. Rodríguez *et al.* [41] and Scharf and Koch [42] agree that the Scrum Master should lead teams into applying Scrum practices, but propose that students with advanced knowledge of the agile philosophy assume this role. Unfortunately, most agile practices require students to have previous experience and to spend time working together, which is not feasible in a first project-based course.

Another option for supporting students during these learning experiences is implementing instances of peer-assessment [24], [43], where students anonymously evaluate the performance of their teammates and provide feedback to help their peers overcome conflicts and limitations. This approach has shown good results and is suitable for use in disciplined developments. However, the comments given to students are mostly based on team members' personal experiences, and may not represent overall team consensus. Moreover, that feedback does not always focus on the key aspects of the team dynamics, due to students' lack of expertise in identifying these aspects [24]. Similar limitations can be also found in self-assessment [44], [45].

Selecting a peer-assessment instrument to support these experiences is also a challenge, since their suitability depends on several project features, such as the team structure (peer-to-peer or based on roles), the software process structure (agile or disciplined), the development scenario (collocated, partially distributed, distributed) and cultural aspects of the team members (individualists, collectivists or mixed). Although several instruments have been proposed for assessing team performance in academic scenarios [39], [46], [47], few of them were specifically created to assess software teams [11], [48], [49], and only the proposal of Silvestre *et al.* [43], [50] considers all the features involved in the projects addressed in this study scenario. Therefore, this proposal is used to inform the quality of teamwork of teams using RWM or not.

On the other hand, instructors frequently use formal course checkpoints to provide feedback to students, typically through Formal Technical Reviews [51]. Although useful, time constraints usually make these reviews superficial, focused on the work products, and with little to no space for reflecting on what went right or wrong in the project. Moreover, students are not given suggestions as early as they should be, since checkpoints are sporadic.

These previous works put in evidence the need to count on instructional methods that help students take advantage of these experiences when a disciplined software strategy is used to guide the running of the project. In this sense, the RWM method adapts general monitoring concepts to software engineering education and uses team reflection during weekly meetings for self-evaluating the project dynamics. Thus, students can learn collaboratively, based on their own experiences.

## III. Context of Use

The Reflexive Weekly Monitoring method was conceived to monitor development teams in a software engineering undergraduate course (CC5401 – Software Engineering II) taught at Universidad de Chile, Chile. The course corresponds to the ninth of the eleven semesters of the Computer

Science program. It is the first project-based course in the curriculum where students have to apply, in a real practical experience, the theoretical knowledge acquired in the previous software engineering courses. This course involves two lectures and one discussion session facilitated by the teaching assistant every week. Of those enrolled in the course, only 20-30% had already worked in the software industry as programmers, either in internships or part time jobs, so according to Robillard and Dulipovici [13] these students could be classified as "inexperienced developers".

The course uses project-based learning as its instructional approach, since is a proven strategy to engage students in authentic real world tasks [52], [53]. The course objective is to familiarize students with the dynamics of developing software in teams and managing these projects while interacting with clients and users. The teams have to work on their projects for twelve weeks, and at the end of which they must deploy the solution.

Every semester the instructor arranges development teams composed of five to seven students and assigns them a project. Typically, they develop Web-based information systems that help address a problem for some unit of the university. These projects involve real users and clients, with their goal and scope having been previously negotiated between the instructor and clients. However, each team is responsible for reaching a formal agreement with its clients and users about the project's scope and how to prioritize and develop its functionality. After this negotiation, both parties sign a document that specifies the software products and deadlines agreed upon.

To maintain similarity across all the software projects under development, all teams must implement their product using a PHP Model-View-Controller framework of their choice, and must also select either a MySQL or PostgreSQL database engine. These technological restrictions make it easier to compare the teams' productivity.

Students are also expected to follow an adaptation of the Simple Software Process (SSP) model [54] to guide their project development. This process is disciplined, according to of Robillard's definition [55], since it has a clear definition of activities, milestones and roles, and also provides templates for the artifacts that the students are to create.

The SSP model involves two increments and eight checkpoints, scheduled by the instructor during the kick-off meeting for all projects. In each checkpoint, an intermediate software artifact – for example, the requirements specification, the software design, or a software prototype - is delivered and evaluated. These checkpoints, open to any participating team, allow the instructor to assess the current status of the product and the project, but not the team internal dynamics or the individual team members' performance. In this sense, the RWM method complements the tracking done by the instructor.

Over twelve weeks of the project, students work in a loosely-coupled way to reach the project goals. Attendance at lectures is not mandatory under university rules, but at least half of the team is expected to attend every weekly monitoring session. These meetings last for an hour, and their frequency is feasible for students while they are still taking courses.

All teams must use the SRM (Software Requirement Manager) tool [54], a shared repository of the projects' work products. The project's resources are only accessible to people working on that project. The SRM also allows students to store and share work products considered in the software process (e.g., user and software requirements, components design and test cases), generate their documentation on-demand based on the information stored in the system, and keep track of the work products' evolution. This tool also reports, on-demand, the status of a project according to the information stored in the system (e.g., fulfilled and unfulfilled items, and committed deadlines). Students and lecturer materials, templates, reading materials, tasks submissions, evaluations and final course evaluation are managed by an e-portfolio tool (U-Cursos), like that reported by Macías [26].

After each iteration, a major software product increment is delivered, and the individual and team performance is evaluated by a peer-assessment that considers both quantitative and qualitative results [43], and that gives feedback to the students.

## IV. Reflexive Weekly Monitoring

The RWM method is led by monitors who perform weekly monitoring sessions with a software development team to facilitate members' understanding of their own performance and project status. The monitor is not a coach, project manager, or Scrum master. Coaches suggest corrective actions and improvements [29], project managers assign tasks, and Scrum Masters lead teams in applying practices and help making decisions [41], but the monitor is an agent who promotes and facilitates the team members' self-reflection, and uses that mechanism to help them find their own solutions to their individual and team problems.

Although the monitor is not responsible for team's activities, he/she can raise warnings when the team is missing critical issues that require a quick response. When students need an external opinion on their diagnosis of the project status, the monitor can provide general feedback about the course of action to be followed, but it is the students alone who are responsible for the project and the final product.

The weekly RWM sessions take place throughout the project, although they are more useful during the early stages. Students should not perceive the sessions as an evaluation, but rather as a formative activity that allows them to improve individually and as a team, based on their previous experiences and without direct consequences on their course grade. Therefore the course instructor and the teaching assistants do not participate in the monitoring activities.

People taking the monitor role do not need significant professional expertise, the projects are usually quite small and are not complex. Similarly, the monitoring method is not complex either and is rigorously defined, so little experience is needed to follow it. Graduate students can assume this role.

Weekly sessions allow monitors to maintain awareness of the projects and teams, recall previous sessions, and facilitate the early identification of risks in the project and the team. This meeting frequency is not intended to overwhelm students or monitors. These sessions involve three sequential

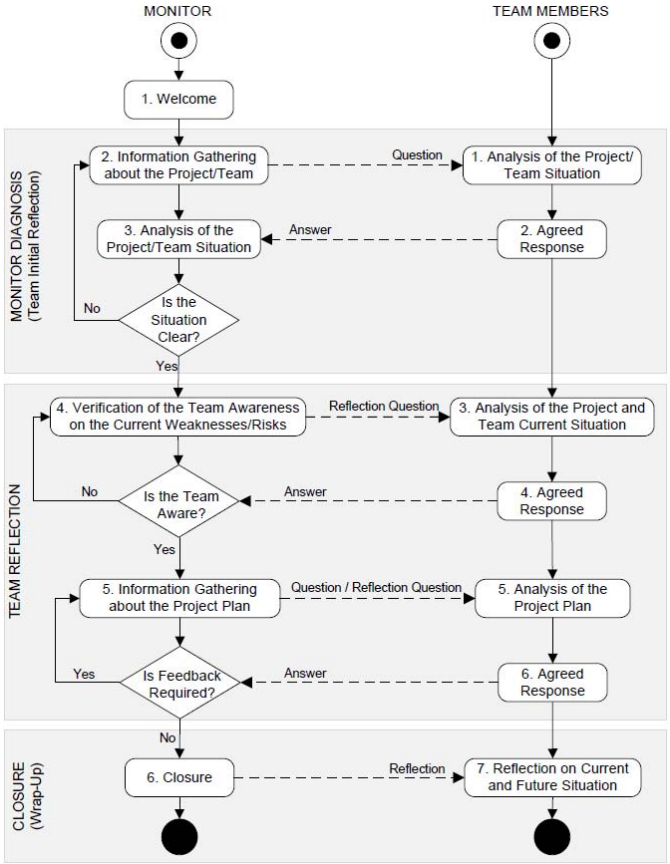


Fig. 1. Structure of a RWM Session.

steps (Fig. 1): *monitor diagnosis*, *team reflection*, and *closure*. This guide helps monitors to conduct the sessions in consistently, and lets students know what to expect. The solid lines in Fig. 1 show the sequence of steps followed by each role during the session, and the dashed lines show the floor control shared between roles. Some activities, like the first done by the team members (that is, the analysis of the project/team situation), involve a meeting that the involved actors (in this case, the team members) must attend, where they receive the floor control from the monitor to start the activity.

### A. Monitor Diagnosis

In each session, the monitor gathers information about the project and team status by asking team members simple questions that require direct answers, such as: "Has everybody accomplished the assigned tasks?", "Is our project on schedule?", "Are we going to be able to accomplish the deadlines?", "Is our team working well?" The monitor, although not part of the team, formulate the questions as though from a team member, to make students feel more comfortable, and not judged, reinforcing the idea of monitoring a formative activity.

The answers to the questions, which must be agreed upon between the team members, are used to make an initial diagnosis of the project progress, their commitment, and the perceived quality of the teamwork. Achieving consensus pushes students to analyze various opinions and points of views, and helps them visualize the current status of the team and project. The monitor can ask additional questions to reach an initial diagnosis, which is usually adjusted during the next steps of the monitoring session. The monitor diagnosis workflow is depicted in the upper part of Figure 1.

### B. Team Reflection

The *team reflection* is designed to make students reflect on possible causes of the issues identified in the team and project diagnosis conducted in the previous step.

The monitor leads the team in a reflexive activity based on questions and answers that focus on the weaknesses and risks of the performed work, project plan, and the attitudes of team members, to arrive at "reflection-in-action" [57], a technique that allows a reshaping of what the students are working on while they are working on it.

### C. Data Analysis

Typical questions asked are: "Why are we delayed?", "What are our main risks/weaknesses?", "How can we address these risks/weaknesses?", "How can we improve our team performance?", and "What can we do to reach the deadlines?"

Answering these questions requires reflection and agreement among teammates. Typically, students only understand the situation when trying to answer these questions. Therefore, this stage is usually the most important part of the process. The monitor has to use his/her own criteria to keep the monitoring session within certain time limits, typically 30 to 40 minutes, so should determine the most relevant questions in each session, and prioritize them.

At the end of this second step the monitor asks students about the project plan to make them reflect on its suitability for addressing the remaining project goals. Eventually, the monitor can offer indirect feedback, which will be discussed during the next monitoring session. This step of workflow is shown in the middle of Figure 1.

### D. Closure

The last step is *closure*, a wrap-up presented by the development team that highlights up to three priority actions to be addressed during next week. The closure step acts as a trigger to create short-term tasks and assignments. If the RWM sessions are applied in long projects, the closure should also consider priority actions for the mid-term. It is recommended that the team self-organizes to address the tasks identified. And thus, determines how they would be carried out. The self-organization requires reflection in and on-action [56] to determine the next steps. This process design is inspired by the guidelines given by Schön [57].

Immediately after the session, the monitor writes a session record that students, teaching assistants, and the instructor do not see until after the end of the course. In the record, the monitor specifies the diagnosis of the project and the team, the actions to be taken for the next week, and optionally some open comments that help keep track of the status of all projects and teams. The team diagnosis includes its engagement with the project, and the level of teamwork and control shown over the development activities. The project diagnosis includes its

state of advancement and risk, and also the appropriateness of the plan to reach the goals.

## V. Evaluation

The effectiveness of the RWM monitoring method on undergraduate software engineering students was studied through a mixed-methods evaluation. Following a between-groups experimental design, a sample of student teams enrolled in the project course described in Section IV were recruited to participate in the study.

### A. Participants

Thirty-two teams (205 students) participated in the study over nine consecutive semesters, from spring 2011 to spring 2015. Each student was enrolled in the course for only one semester, hence being assigned to just one team. Sixty-five percent of teams were formed only by male students whereas 35% of teams were mixed. From the latter, 22% had one female student (seven teams) and 13% had two female students (three teams). All participants were of similar age, ranging between 22 and 25 years old.

Adhering to the course instructional approach, each team was assigned a project to be independently addressed over a period 12-week period. All enrolled students belonged to the same undergraduate program, which is rigid and comprises mostly mandatory courses, and thus they had comparable technical skills at the beginning of the project. Furthermore, they had similar prior experience in software development since they all had completed their first professional internship that lasts for one month. The course instructor and the leading teaching assistant were the same for the whole study period, and neither participated in the monitoring sessions, since their involvement could possibly influence student behavior and final grades in the course.

Teams were randomly assigned to the experimental condition, i.e., being monitored following the RWM method. At the end of the study, eighteen teams (112 students) were monitored while the other fourteen (93 students) were not monitored. Non-monitored teams served as the control group and did not show significantly lower performance—measured in terms of the final grade obtained at the end of the course—as compared to non-monitored teams not participating in the study (i.e., those before the observation period). Comparing the grades of nine semesters prior to this empirical evaluation, the average course grade was 5.62 with a standard deviation of 0.882, whereas non-monitored teams that participated in the experiment had an average course grade of 5.51 with a standard deviation of 0.993.

For the monitored teams, a group of graduate students in computer science assumed the role of monitors. Some were teaching assistants in other introductory computer science undergraduate courses at the university and were recruited based on their prior experience in working and leading software development teams. New monitors were required to complete a brief instruction period on how to conduct the monitoring sessions in which they were trained in the steps, goals, and scope of the RWM method. To ensure consistency across monitors, they were observed by the first author when delivering their first two or three sessions.

### B. Data Collection

To compare the experimental group (monitored teams) to the control group (non-monitored teams), the study used the information gathered using the regular course tracking instruments - the peer assessment conducted at the end of each increment and a statistical report provided by the requirements tracking tool (SRM) [54] used by the students in their projects.

For the anonymous peer-assessment students used the instrument proposed by Silvestre *et al.* [43], [50], designed specifically to evaluate software team performance in the scenario considered in this study; i.e., software teams of Latin American undergraduate students following a disciplined software process in a loosely-coupled way to develop a Web information system. This instrument, a 5-point Likert scale questionnaire, evaluates seven aspects of teammates (engagement, sense of belonging to a team, responsibility, initiative, communication, coordination, and desire to improve) using several weightings that are the result of an empirical study. Students had to indicate their level of agreement or disagreement with each statement of a dimension of teamwork with one of the following options: 1-never, 2-almost never, 3-sometimes, 4-regularly, and 5-always. Analysis of the responses made it was possible to study the validity of hypotheses 1 and 2. The instrument also elicits open comments about the strengths and weaknesses of the teammate being evaluated. Each aspect is represented through one or more sentences describing an expected attitude of a teammate. For instance, a description of the coordination aspect states: *"He/she fulfills his/her assignments properly, working transparently and coordinated with other peers"*, and that evaluating the sense of belonging to a team is: *"He/she assumes the project as a team effort, providing support to peers in the project tasks"*. The validity of this instrument was assessed with a representative sample of the study population [50]. Opinions reported on the peer-assessment items were found to be highly reliable, by calculating the associated Cronbach's alpha for each subscale ($\alpha = 0.86$ for coordination and $\alpha = 0.91$ for sense of belonging to a team).

Students used the SRM tool [54] to manage the scope and current status of their projects according to their requirements. As an objective measure for effectiveness (i.e., whether a piece of software was actually deployed and used to solve a particular problem) and productivity (i.e., the number of software requirements met by the team), the information provided by SRM was used to identify the size and completeness of each project after the second increment. In particular, SRM lists the software requirements for all projects, and a status code indicating its final level of completion (fulfilled, unfulfilled or ambiguous). The information stored in SRM can be considered valid and reliable since it is the result of the last software inspection performed by the instructor and teaching assistants just before the project closure. The effort needed to address each functional requirement was estimated using the RESC (Raw Estimation based on Standard Components)

method [58], which is an adaptation of PROBE (Proxy-Based Estimating) [59], specifying the development effort in terms of function points (FP). RESC allows the development effort for new components to be estimated using a set of standard components (e.g., CRUDs, reports, and Web menus) as reference points, whose development effort is already known for that context. Therefore, this effort estimation is based on historical information gathered from the same work context. To minimize bias on the effort estimation, this was done individually and independently by the second author and two other experienced software engineers, based on historical information for that work context and the software requirements for the project. The absolute difference between these estimations was of 14% in average ($max = 18\%$, $min = 5\%$, $med = 10\%$). The information provided by SRM allows evaluation of the validity of hypotheses 3 and 4.

### C. Data Analysis

A two-tailed independent samples t-test was performed over the empirical data to study the validity of the stated hypotheses. All the statistical analyses were done using SPSS version 23.

Hypotheses 1 and 2 were evaluated using the data obtained from the peer-assessment. The Cronbach's alpha coefficient was computed for the answers provided by the participants in each team, to analyze the internal consistency of the peer-assessment. For hypotheses 3 and 4, the criticality level of the requirements was used to determine team effectiveness (H3), and their level of completeness and the effort required to address them was used to determine team productivity (H4).

Finally, to cross-check the results obtained from the peer-assessment and SRM, the values observed were triangulated with qualitative data from handwritten records of the monitoring sessions and from comments on student evaluations of courses and instructors that the university applies twice per semester (at the middle and end of the course). At the end of the semester all the handwritten records were delivered for analysis to the instructor who performed an open, axial, and selective coding on the data, and then grouped the emerging themes in affinity diagrams mapping the study hypotheses.

## VI. Results

Choosing a significance level of $\alpha = 0.05$ for all hypotheses, the null hypotheses was rejected for H1, H2, and H3, while rejection of the null hypothesis for H4 failed. Table I summarizes the observed values for the study sample. Next, the obtained results and a discussion on the validity of the hypotheses are presented.

TABLE I
Coordination, Sense of Belonging to a Team, Effectiveness, and Productivity of Team Members

| | Team | Coordination | Sense of Belonging to a Team | % of Fulfilled Critical Requirements | FP Fulfilled per Team Member | Cronbach's Alpha |
|---|---|---|---|---|---|---|
| Non-Monitored | Team 1 | 4.43 | 4.32 | 100% | 17.79 | 0.82 |
| | Team 2 | 3.54 | 3.68 | 52% | 9.19 | 0.81 |
| | Team 3 | 4.36 | 4.46 | 91% | 10.62 | 0.93 |
| | Team 4 | 4.28 | 4.03 | 100% | 23.72 | 0.89 |
| | Team 5 | 4.38 | 4.52 | 100% | 13.51 | 0.99 |
| | Team 5 | 3.74 | 4.19 | 100% | 14.13 | 0.88 |
| | Team 7 | 4.14 | 4.48 | 91% | 9.59 | 0.89 |
| | Team 8 | 4.49 | 4.40 | 100% | 19.08 | 0.91 |
| | Team 9 | 3.87 | 4.07 | 73% | 18.68 | 0.92 |
| | Team 10 | 4.06 | 4.36 | 90% | 15.11 | 0.84 |
| | Team 11 | 4.37 | 4.33 | 69% | 14.38 | 1.00 |
| | Team 12 | 4.47 | 4.41 | 59% | 19.01 | 0.93 |
| | Team 13 | 4.31 | 4.19 | 92% | 10.17 | 0.84 |
| | Team 14 | 4.08 | 4.22 | 86% | 8.11 | 0.86 |
| Monitored | Team 1 | 4.42 | 4.62 | 100% | 13.02 | 0.82 |
| | Team 2 | 4.32 | 4.28 | 100% | 23.52 | 0.89 |
| | Team 3 | 4.79 | 4.79 | 100% | 8.33 | 0.80 |
| | Team 4 | 4.53 | 4.80 | 100% | 9.97 | 0.93 |
| | Team 5 | 4.72 | 4.35 | 100% | 16.28 | 0.90 |
| | Team 6 | 4.69 | 4.62 | 100% | 9.54 | 0.83 |
| | Team 7 | 4.63 | 4.53 | 100% | 14.37 | 0.89 |
| | Team 8 | 4.77 | 4.88 | 100% | 18.58 | 0.82 |
| | Team 9 | 4.54 | 4.64 | 100% | 16.39 | 0.99 |
| | Team 10 | 4.52 | 4.52 | 100% | 12.81 | 0.82 |
| | Team 11 | 4.78 | 4.90 | 100% | 13.63 | 0.93 |
| | Team 12 | 4.73 | 4.63 | 100% | 12.92 | 0.91 |
| | Team 13 | 4.72 | 4.62 | 93% | 10.54 | 0.79 |
| | Team 14 | 4.69 | 4.68 | 100% | 12.63 | 0.83 |
| | Team 15 | 4.62 | 4.53 | 100% | 9.25 | 0.89 |
| | Team 16 | 4.74 | 4.78 | 100% | 9.14 | 0.92 |
| | Team 17 | 4.58 | 4.53 | 100% | 14.78 | 0.80 |
| | Team 18 | 4.50 | 4.62 | 100% | 9.73 | 0.81 |

### A. Coordination

Monitored teams ($M = 4.61$, $SD = 0.62$) reported being more coordinated than non-monitored teams ($M = 4.17$, $SD = 1.02$), $t(1796) = 11.222$, $p = 0.00001$, $d = 0.52$, 95% CI = [0.3635, 0.5175].

The comments included in the peer-assessments indicate some coordination limitations for both types of teams after the first increment. However, after the second increment, these limitations only appear in non-monitored teams.

This matches the observations both of monitors, who reported that "There seems to be an improvement in the team members' coordination after the first increment", and of participants in monitored teams, who stated: *"The monitoring sessions showed us the need for coordinatin*g our activities. Unfortunately, we understood at the very last stage of the project", and also that "We started out uncoordinated, but this situation changed after each [monitoring] session. I feel that we finished working as a team". Therefore, the results obtained support the first hypothesis (H1: the RWM method positively impacts coordination between team members).

### B. Sense of Belonging to a Team

Monitored teams ($M = 4.60$, $SD = 0.63$) showed a better sense of belonging than non-monitored teams ($M = 4.22$,

$SD = 1.06$) did, $t(1796) = 9.614$, $p = 0.00001$, $d = 0.45$, 95% $CI = [0.3057, 0.4624]$.

Monitors identified a significant effort made by students to come together as a team during the first increment: *"Most [monitored] students work hard during the first increment to become part of their teams, and most of them seem to succeed"*. The sessions seem to motivate the students to make an effort to become a team.

In that respect, students in monitored teams mentioned: *"The sessions help us realize that people can contribute in several ways, and this understanding made us a great team"*, and *"The weekly discussions about team limitations allowed us to overcome them"*.

This was also recognized by students in non-monitored teams, some of whom felt that the lack of a weekly monitoring session had harmed them: *"We finished acting as a group of developers, because we were not monitored (...). In this sense, the monitoring sessions favored the other teams"*.

In summary, the results obtained support the second hypothesis (H2: *RWM increases the sense of belonging to a team*), and show that members of monitored teams tend to be more motivated to become a team and more satisfied with the overall experience than those in non-monitored teams.

### C. Effectiveness

Monitored team members ($M = 99\%$, $SD = 1\%$) work more effectively than members of non-monitored teams ($M = 86\%$, $SD = 16\%$), $t(30) = 3.570$, $p = 0.001$, $d = 1.18$, 95% $CI = [5.85\%, 21.51\%]$. Here, effectiveness is measured in terms of the coverage of critical requirements met in the software project.

A requirement is considered "critical" if it must be addressed to put the software into production (i.e., if it is mandatory). The criticality of each requirement is defined by the development team using the SRM tool [54]. According to Table I, 94% of the monitored teams and 36% of the non-monitored teams were able to address all critical requirements, making thus possible to deploy the software.

This finding was corroborated by the stated opinions of students and monitors. For instance, a monitor reported: *"Monitoring sessions show the students what is mandatory in the project. Therefore, the problem is that they work only on these features"*. A member of a monitored team indicated: *"The sessions helped us to quickly identify the core [i.e., the set of mandatory requirements]. After that, the development became more manageable"*.

Therefore, the study results support the third hypothesis (H3: *RWM makes the students work more effectively*). In other words, RWM sessions helped developers keep the focus on the critical aspects of the product, resulting in a more effective development.

### D. Productivity

Productivity—measured as the amount of FP fulfilled per team member in each software project—in monitored teams ($M = 13.04$, $SD = 3.91$) and non-monitored teams ($M = 14.51$, $SD = 4.66$) showed little difference, $t(30) = 0.969$, $p = 0.340$, $d = 0.340$, 95% $CI = [-4.563, 1.626]$. Monitored teams were not more productive than non-monitored teams, contradicting the fourth hypothesis. However, this aspect seems to require further analysis. All monitors observed how students reduced their development effort when they felt that their project is under control. For instance, a monitor stated: *"During the sessions I am as skeptical as I can be of the project's success, to avoid that students slacking off in the middle of the term"*. Students change from an active attitude that shows commitment with the project, to a speculative attitude that shows commitment with the project, to a speculative attitude where they work just to accomplish the goals. This observation resonates with the "apprentice attitude" [19], [20] discussed in related literature.

This change of attitude made it difficult to determine the real impact of the monitoring activities on students' productivity. This attitude also affects their interest in reflecting on their own activities. Some students believed that team reflection became optional once they were in control of the project: *"The sessions should not be required any longer if we have the project under control. We know that we are going to succeed; doing it earlier does not make any difference"*.

## VII. Discussion

The evaluation results have both similarities with, and differences to prior findings reported in the literature. Similar to experiences using coaches [24], [28], [29] or Scrum masters to support student teams [40]–[42], RWM uses monitors to enhance students' learning experience during software projects. However, monitors do not guide nor examine students, but instead act as facilitators for self-reflection by team members. This probably caused the situation where no tension with monitored students was observed during the evaluation process. Although all these three approaches are useful for helping students work in a coordinate fashion, the use of RWM caused students to realize, at an early stage of the project, the importance of coordinated working and, eventually, of changing their behavior for the benefit of the team. Team grades prior to the experiment and team grades of non-monitored teams (presented in Section VIII) did not differ significantly. Furthermore, the course evaluation criteria were the same during the nine semesters covered in the experiment and in the nine semesters prior to the experiment.

In terms of team effectiveness, again all three approaches show good results. However, the prior literature is not clear as to exactly when in the project the teams supported by coaches or Scrum masters realize that they are in control, nor how their attitudes change after that point. RWM has been shown to be highly effective at helping students identify the mandatory aspects of the project, but this has been counterproductive, since students tend to show an "apprentice" attitude after that point, minimizing the effort they invest in the project and focusing only on the mandatory requirements. In future work, topics such as deployment, application end-user feedback and functionality will be analyzed as complementary measures for evaluating team effectiveness.

In terms of team monitoring capability, all three approaches have been shown to be suitable and able to identify procrastination and free riders. However, RWM achieves these outcomes with teams using a disciplined software process with novice developers, an instructional scenario little explored in the literature. Moreover, monitored teams tend not to feel that they are being observed by someone else. In this sense, this proposal represents an advance of the state-of-the-art. Despite the good results of applying the RWM method, there were dysfunctional teams who did not work well together, and where personal conflicts jeopardized the effort made by the monitor and the instructor to put the team on the right track.

Despite the good reports of using e-portfolios and similar tools as instruments to enhance and monitor these experiences [25]–[27], in RWM it is not mandatory to use a shared repository. The use of these tools is compatible with RWM only if they are not actively used by the instructor to monitor students during the project. In practice, this strategy has shown to promote students' empowerment, commitment, and sense of belonging - attitudes that needed reinforcement in these experiences. This result of the RWM sessions was recognized by members of both monitored and non-monitored teams.

An aspect that has not been deeply explored in the literature yet is the impact these experiences have on the instructional process. At the end of the course the monitors organized their annotations chronologically and wrote a brief diagnosis of the team and project over the whole period, and gave these diagnoses and the comments to the instructor, who indicated that these RWM results helped him determine weaknesses and opportunities to improve the software process that guides these projects. For instance, the monitoring sessions showed that it took students too long to take control of the project. Therefore, the new version of the software process used in the course includes a conception phase before the two increments, which has helped mitigate this limitation. This unexpected result indicates that the RWM method not only helps students to enhance their learning experience, but also helps instructors to improve the process that guides these activities.

## VIII. Threats to Validity

Although the reported study results are valuable, there may be an issue with the quality of the data collected from the peer-assessment. Students are expected to make this assessment fairly, but sometimes they do not evaluate their peers as they should. To mitigate this threat some outlying opinions were removed and the results were corroborated with the monitors' records once the course ended.

The instrument used in the peer-assessment is the result of an empirical study done by Silvestre *et al.* [43], [50], on 12 software teams (74 students) working in a scenario similar to that studied here. The instrument has shown consistently valid results, but has not been evaluated enough as to consider it infallible.

On the other hand, the estimation of function points accomplished by the teams during their projects could be biased, since such an activity is error prone. This threat was mitigated with the help of two external and experienced software engineers who estimated the FP of the students' projects separately, based on historical information of the effort required by students to develop standard components for Web information systems in that course. The differences between the external estimations and those reported in Table I were not significant (maximum difference of 18%) and can be taken as a reasonable variation between the three estimations, which makes thus suitable to compare the development efforts in the students' projects.

Finally, the evaluation process captured the reality of a particular software engineering project course, and that may be specific to the Department of Computer Science of the Engineering School at the Universidad de Chile. Therefore, conducting a similar experience in other universities may not yield the same results. Replicating the experience in a different scenario would help assess to what extent these results could be generalized.

## IX. Conclusion

This paper described the RWM method, designed to help students of software engineering project courses to reflect on their individual and team performance when using a disciplined software process. Thus, this method enhances the students' learning experiences during these instructional activities.

After applying RWM in 18 out of 32 teams, the results obtained indicate that the monitoring sessions helped students increase their coordination, effectiveness, and sense of belonging to a team, but did not necessarily help their productivity, which was negatively affected by the apprentice attitude of the students. Given the success of this study, the RWM method has been incorporated as a standard practice in the instructional approach since 2016.

These sessions also allowed monitored teams to make a more accurate diagnosis of their projects and team status, since they had a weekly space for reflection. This allowed them to quickly react in case of problems, detect risks early, and learn from past experiences. The internal work climate was also favored by these weekly discussions.

Conversely, many non-monitored teams failed in determining what team and project challenges had to be addressed in their development projects. The lack of a formal reflection space may be why they were slow to identify these, and thus why most teams tended to react late.

RWM' benefits are similar to those of other approaches, like the use of coaches, Scrum masters, and e-portfolios. However, RWM sums up the benefits of all of these using disciplined processes and without generating tension between students. Instructors can also take advantage of the RWM monitoring information to improve the software process used to guide these experiences.

The RWM method was designed for use in software engineering courses, but this does not prevent its use in other engineering project courses whose students work in teams using a disciplined process to solve a problem, although, the results shown in this study may not be representative of other scenarios.

A following step in this research is to perform a more in-depth study of students' productivity, to understand their variability and to confirm the findings of this study. Based on the current study results, it could be hypothesized that team productivity is more related to the size of the system core and to student engagement, than to the development capability of team members.

## REFERENCES


[1] A. Begel and B. Simon, "Novice software developers, all over again," in *Proc. 4th Int. Workshop Comput. Educ. Res. (ICER)*, New York, NY, USA, 2008, pp. 3–14.

[2] A. M. Moreno, M.-I. Sánchez-Segura, F. Medina-Dominguez, and L. Carvajal, "Balancing software engineering education and industrial needs," *J. Syst. Softw.*, vol. 85, no. 7, pp. 1607–1620, Jul. 2012.

[3] Y. Sedelmaier and D. Landes, "Active and inductive learning in software engineering education," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng. (ICSE)*, Florence, Italy, 2015, pp. 418–427.

[4] J. E. Tomayko and O. Hazzan, *Human Aspects of Software Engineering* (Electrical and Computer Engineering Series). Rockland, MA, USA: Charles River Media, 2004.

[5] M. John, F. Maurer, and B. Tessem, "Human and social factors of software engineering: Workshop summary," *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–6, 2005.

[6] D. Broman, K. Sandahl, and M. A. Baker, "The company approach to software engineering project courses," *IEEE Trans. Educ.*, vol. 55, no. 4, pp. 445–452, Nov. 2012.

[7] M. Gehrke *et al.*, "Reporting about industrial strength software engineering courses for undergraduates," in *Proc. 22rd Int. Conf. Softw. Eng. (ICSE)*, Orlando, FL, USA, May 2002, pp. 395–405.

[8] Y. Sedelmaier and D. Landes, "Practicing soft skills in software engineering: A project-based didactical approach," in *Overcoming Challenges in Software Engineering Education: Delivering Non-Technical Knowledge and Skills: Delivering Non-Technical Knowledge and Skills*. Hershey, PA, USA: IGI Glob., 2014, p. 161.

[9] J. D. Tvedt, R. Tesoriero, and K. A. Gary, "The software factory: Combining undergraduate computer science and software engineering education," in *Proc. 23rd Int. Conf. Softw. Eng. (ICSE)*, Toronto, ON, Canada, May 2001, pp. 633–642.

[10] T. Markham, "Project based learning: A bridge just far enough," *Teacher Librarian*, vol. 39, no. 2, pp. 38–42, 2011.

[11] D. Dietsch, A. Podelski, J. Nam, P. M. Papadopoulos, and M. Schäf, "Monitoring student activity in collaborative software development," *CoRR*, vol. abs/1305.0787, 2013. [Online]. Available: http://dblp.uni-trier.de/rec/bibtex/journals/corr/abs-1305-0787

[12] H. Song, G. Si, L. Yang, H. Liang, and L. Zhang, "Using project-based learning and collaborative learning in software engineering talent cultivation," in *Proc. 10th IEEE Int. Conf. Trust Security Privacy Comput. Commun. (TrustCom)*, Changsha, China, 2011, pp. 1288–1293.

[13] P. N. Robillard and M. Dulipovici, "Teaching agile versus disciplined processes," *Int. J. Eng. Educ.*, vol. 24, no. 4, pp. 671–680, 2008.

[14] Y. Sedelmaier and D. Landes, "Software engineering body of skills (SWEBOS)," in *Proc. Glob. Eng. Educ. Conf. (EDUCON)*, Istanbul, Turkey, 2014, pp. 395–401.

[15] L. B. Sherrell and S. G. Shiva, "Will earlier projects plus a disciplined process enforce SE principles throughout the CS curriculum?" in *Proc. 27th Int. Conf. Softw. Eng. (ICSE)*, St. Louis, MO, USA, 2005, pp. 619–620.

[16] V. Mahniè, "Scrum in software engineering courses: An outline of the literature," *Glob. J. Eng. Educ.*, vol. 17, no. 2, pp. 77–83, 2015.

[17] M. Hoegl and H. G. Gemuenden, "Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence," *Org. Sci.*, vol. 12, no. 4, pp. 435–449, 2001.

[18] M. Marques, "A prescriptive software process for academic scenarios," Ph.D. dissertation, Dept. Comput. Sci., Universidad de Chile, Santiago, Chile, 2017.

[19] B. W. Boehm and D. Port, "Educating software engineering students to manage risk," in *Proc. 23rd Int. Conf. Softw. Eng. (ICSE)*, Toronto, ON, Canada, May 2001, pp. 591–600.

[20] J. C. Schlimmer, J. B. Fletcher, and L. A. Hermens, "Team-oriented software practicum," *IEEE Trans. Educ.*, vol. 37, no. 2, pp. 212–220, May 1994.

[21] J. Chen, H. Lu, L. An, and Y. Zhou, "Exploring teaching methods in software engineering education," in *Proc. 4th Int. Conf. Comput. Sci. Educ. (ICCSE)*, Nanning, China, 2009, pp. 1733–1738.

[22] A. Heredia, R. Colomo-Palacios, and A. Amescua-Seco, "A systematic mapping study on software process education," in *Proc. SPETP SPICE*, 2015, pp. 7–17.

[23] M. R. Marques, A. Quispe, and S. F. Ochoa, "A systematic mapping study on practical approaches to teaching software engineering," in *Proc. 44th Annu. Frontiers Educ. Conf. (FIE)*, Madrid, Spain, 2014, pp. 1–8.

[24] Y.-P. Cheng and J. M.-C. Lin, "A constrained and guided approach for managing software engineering course projects," *IEEE Trans. Educ.*, vol. 53, no. 3, pp. 430–436, Aug. 2010.

[25] N. L. Carroll, L. Markauskaite, and R. A. Calvo, "E-portfolios for developing transferable skills in a freshman engineering course," *IEEE Trans. Educ.*, vol. 50, no. 4, pp. 360–366, Nov. 2007.

[26] J. A. Macías, "Enhancing project-based learning in software engineering lab teaching through an E-portfolio approach," *IEEE Trans. Educ.*, vol. 55, no. 4, pp. 502–507, Nov. 2012.

[27] V. Mahniè and A. Èasar, "A computerized support tool for conducting a scrum-based software engineering capstone course," *Int. J. Eng. Educ.*, vol. 32, pp. 278–293, Jan. 2016.

[28] R. Bareiss and M. Griss, "A story-centered, learn-by-doing approach to software engineering education," *ACM SIGCSE Bull.*, vol. 40, no. 1, pp. 221–225, Mar. 2008.

[29] G. Rodríguez, A. Soria, and M. Campo, "Measuring the impact of agile coaching on students' performance," *IEEE Trans. Educ.,* vol. 59, no. 3, pp. 202–209, Aug. 2016.

[30] B. A. D. Jong and T. Elfring, "How does trust affect the performance of ongoing teams? The mediating role of reflexivity, monitoring, and effort," *Acad. Manag. J.*, vol. 53, no. 3, pp. 535–549, 2010.

[31] G. R. Jones, "Task visibility, free riding, and shirking: Explaining the effect of structure and technology on employee behavior," *Acad. Manag. Rev.*, vol. 9, no. 4, pp. 684–695, 1984.

[32] R. Rico, M. Sánchez-Manzanares, F. Gil, and C. Gibson, "Team implicit coordination processes: A team knowledge based approach," *Acad. Manag. Rev.*, vol. 33, no. 1, pp. 163–184, 2008.

[33] M. A. Marks and F. J. Panzer, "The influence of team monitoring on team processes and performance," *Human Perform.*, vol. 17, no. 1, pp. 25–41, 2004.

[34] V. Garousi, "Applying peer reviews in software engineering education: An experiment and lessons learned," *IEEE Trans. Educ.*, vol. 53, no. 2, pp. 182–193, May 2010.

[35] J. S. Collofello and M. Hart, "Monitoring team progress in a software engineering project class," in *Proc. 29th Annu. Frontiers Educ. Conf. (FIE)*, vol. 1. Piscataway, NJ, USA, 1999, pp. 11B4/7–11B410. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/839226/

[36] M. A. Bakar, N. Jailani, Z. Shukur, and N. F. M. Yatim, "Final year supervision management system as a tool for monitoring computer science projects," *Proc. Soc. Behav. Sci.*, vol. 18, pp. 273–281, Jan. 2011.

[37] C. Jones, "Using subversion as an aid in evaluating individuals working on a group coding project," *J. Comput. Sci. Colleges*, vol. 25, no. 3, pp. 18–23, Jan. 2010.

[38] W. Poncin, A. Serebrenik, and M. van den Brand, "Mining student capstone projects with FRASR and ProM," in *Proc. ACM Int. Conf. Companion Object Orient. Program. Syst. Lang. Appl. Companion*, Portland, OR, USA, 2011, pp. 87–96.

[39] J. Kay, N. Maisonneuve, K. Yacef, and O. Zaïane, "Mining patterns of events in students' teamwork data," in *Proc. Workshop Educ. Data Min. 8th Int. Conf. Intell. Tutoring Syst. (ITS)*, Jun. 2006, pp. 45–52.

[40] V. Mahnic, "A capstone course on agile software development using scrum," *IEEE Trans. Educ.*, vol. 55, no. 1, pp. 99–106, Feb. 2012.

[41] G. Rodríguez, Á. Soria, and M. Campo, "Virtual scrum: A teaching aid to introduce undergraduate software engineering students to scrum," *Comput. Appl. Eng. Educ.*, vol. 23, no. 1, pp. 147–156, 2015.

[42] A. Scharf and A. Koch, "Scrum in a software engineering course: An in-depth praxis report," in *Proc. 26th Int. Conf. Softw. Eng. Educ. Train. (CSEE T)*, San Francisco, CA, USA, 2013, pp. 159–168.

[43] L. Silvestre, S. F. Ochoa, and M. Marques, "Understanding the design of software development teams for academic scenarios," in *Proc. 34th Int. Conf. Chil. Comput. Sci. Soc. (SCCC)*, Santiago, Chile, 2015, pp. 1–6.

[44] J. H. Hayes, T. C. Lethbridge, and D. Port, "Evaluating individual contribution toward group software engineering projects," in *Proc. 25th Int. Conf. Softw. Eng.*, May 2003, Portland, OR, USA, 2003, pp. 622–627.

[45] G. G. Mitchell and J. D. Delaney, "An assessment strategy to determine learning outcomes in a software engineering problem-based learning course," *Int. J. Eng. Educ.*, vol. 20, no. 3, pp. 494–502, 2004.
[46] M. W. Ohland *et al.*, "The comprehensive assessment of team member effectiveness: Development of a behaviorally anchored rating scale for self and peer evaluation," *Acad. Manag. Learn. Educ.*, vol. 11, no. 4, pp. 609–630, 2012.
[47] C. Hastie, K. Fahy, and J. Parratt, "The development of a rubric for peer assessment of individual teamwork skills in undergraduate midwifery students," *Women Birth*, vol. 27, no. 3, pp. 220–226, 2014.
[48] P. Fernandes, A. Sales, A. R. Santos, and T. Webber, "Performance evaluation of software development teams: A practical case study," *Electron. Notes Theor. Comput. Sci.*, vol. 275, pp. 73–92, Sep. 2011.
[49] S. Huang *et al.*, "Toward objective and quantitative assessment and prediction of teamwork effectiveness in software engineering courses," *SIGSOFT Softw. Eng. Notes*, vol. 38, no. 1, pp. 7–9, Jan. 2013.
[50] L. Silvestre, "Design of software teams in academic scenarios (in Spanish)," M.S. thesis, Dept. Comput. Sci., Univ. Chile, Santiago, Chile, 2011.
[51] D. P. Freedman and G. M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products*, 3rd ed. New York, NY, USA: Dorset House, 2000.
[52] L. Johns-Boast and S. Flint, "Simulating industry: An innovative software engineering capstone design course," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, 2013, pp. 1782–1788.
[53] A. Martínez-Monés *et al.*, "Multiple case studies to enhance project-based learning in a computer architecture course," *IEEE Trans. Educ.*, vol. 48, no. 3, pp. 482–489, Aug. 2005.
[54] S. F. Ochoa, J. A. Pino, L. A. Guerrero, and C. A. Collazos, "SSP: A simple software process for small-size software development projects," in *Proc. IFIP 19th World Comput. Congr. 1st Int. Workshop Adv. Softw. Eng. Expanding Front. Softw. Technol.*, Santiago, Chile, 2006, pp. 94–107.
[55] P. N. Robillard, P. Kruchten, and P. d'Astous, "YOOPEEDOO (UPEDU): A process for teaching software process," in *Proc. 14th Conf. Softw. Eng. Educ. Train.*, Charlotte, NC, USA, 2001, pp. 18–26.
[56] S. F. Ochoa, A. Quispe, A. Vergara, and J. A. Pino, "Improving requirements engineering processes in very small software enterprises through the use of a collaborative application," in *Proc. 14th IEEE Int. Conf. Comput. Supported Cooper. Work Design (CSCWD)*, Shanghai, China, Apr. 2010, pp. 116–121.
[57] D. A. Schön, "Teaching artistry through reflection-in-action," in *Educating the Reflective Practitioner*. San Francisco, CA, USA: Jossey-Bass, 1997, pp. 22–40.
[58] M. K. Smith. (2001). *Donald Schön: Learning, Reflection and Change*. Accessed on Aug. 25, 2017. [Online]. Available: http://www.infed.org/thinkers/et-schon.htm
[59] M. A. Marks, J. E. Mathieu, and S. J. Zaccaro, "A temporally based framework and taxonomy of team processes," *Acad. Manag. Rev.*, vol. 26, no. 3, pp. 356–376, 2001.
[60] (May 2007). *También Hay Que sustituir 'Spanish' por 'Spanish' Porque en Inglés va Siempre Con Mayúscula*. [Online]. Available: http://www.puce.edu.ec/sitios/publicaciones/Centro_de_Publicaciones/Revistas/Publicaciones/Revista%2081.pdf#page=125
[61] W. S. Humphrey, *A Discipline for Software Engineering*. Boston, MA, USA: Addison-Wesley, 1995.

**Maíra Marques** received the B.Sc. degree in chemical engineering from the Universidade Federal de São Carlos, Brazil, and the M.Sc. and Ph.D. degrees in computer science from the Universidad de Chile in 2011 and 2017, respectively. Her research interest areas include software processes, collaborative work, and software engineering education. Her current research is on software product lines and software engineering education.

**Sergio F. Ochoa** (M'12) received the Ph.D. degree in computer science from the Pontificia Universidad Católica de Chile. He is an Associate Professor of computer science with Universidad de Chile. His research interests include computer-supported collaborative work, software engineering, educational technology, and mobile and ubiquitous computing. He is a member of ACM and the Chilean Computer Science Society and the steering committee of Latin American and Caribbean collaborative ITC Research Initiative.

**María Cecilia Bastarrica** received the degree in informatics engineering from the Universidad Católica del Uruguay, the M.Sc. degree from Pontificia Universidad Católica de Chile, and the Ph.D. degree in computer science and engineering from the University of Connecticut, USA. She is an Associate Professor with the Department of Computer Science, Universidad de Chile. Her research interests are software engineering, model-driven engineering, and software process modeling and formalization. She is currently engaged in technology transfer projects, specifically in small software enterprises.

**Francisco J. Gutierrez** received the Diplôme d'Ingénieur (equivalent to the M.Sc. degree in engineering) degree from École Centrale de Nantes, France and the Ph.D. degree in computer science from the Universidad de Chile. His research interests are in social computing, human-computer interaction, empirical studies in software engineering, and computer science education. He is currently committed to studying interpersonal communication and cooperative practices of group work as a way to inform the design of new technology to connect people and innovate in educational settings.