University of Memphis

University of Memphis Digital Commons

Electronic Theses and Dissertations

12-7-2022

Scaffolding Strategic Planning and Reflection within Software Engineering Student Team

Murshida Rahman Mouree

Follow this and additional works at: https://digitalcommons.memphis.edu/etd

Recommended Citation

Mouree, Murshida Rahman, "Scaffolding Strategic Planning and Reflection within Software Engineering Student Team" (2022). *Electronic Theses and Dissertations*. 3170.
https://digitalcommons.memphis.edu/etd/3170

This Thesis is brought to you for free and open access by University of Memphis Digital Commons. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of University of Memphis Digital Commons. For more information, please contact khggerty@memphis.edu.

SCAFFOLDING STRATEGIC PLANNING AND REFLECTION WITHIN SOFTWARE ENGINEERING STUDENT TEAMS

by

Murshida Rahman Mouree

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Computer Science

The University of Memphis

December 2022

Copyright©Murshida Rahman Mouree
All rights reserved

ABSTRACT

Mouree, Murshida Rahman. Scaffolding Strategic Planning and Reflection within Software Engineering Student Teams. Major Professor: Dr. Amy Cook.

Strategic planning is essential for a student software engineering team to successfully accomplish a project. Often student software teams struggle with planning, because it is hard for novices to distribute work among teammates, identify tasks, prioritize development tasks, and know how much time they should invest in each task. In other words, students struggle to self-regulate their learning. Prior work in this area typically focuses on an individual, and does not promote iterative reflection and feedback on a collaboratively created plan.

In this paper, we introduce Software Engineering Team Strategic Planning (SETS Planning), a novel strategic planning and reflection technique that helps student teams plan software development tasks. We conducted a pilot test of our planning technique in an undergraduate Software Engineering course over two semesters. We report student perceptions of the experience, how students used and deviated from their plans, and teamwork challenges.

Our findings show that when teams used our planning technique, recognizing and following the development tasks was smoother, students were more organized in team works, and estimating development time was closer to actual time. We also found that when teams use our process, they generate data that reveals the high performers and low performers within a team and helps the TA provide better help to a team.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Students in Software Engineering (SE) development teams often fail to follow a strategic plan. Although task planning can be helpful for expert software developers, amateur students can easily be lost in the complexity of real development work. Students struggle to schedule work according to milestones and accurately estimate how much time they should invest in each task. Working in a group can be tricky because many development tasks interconnect. If students do not understand the underlying development steps, dividing the tasks among group members can be impossible. Without effective planning, students struggle to work in teams and finish work on time. Ultimately, they fail to self-regulate their learning, which is a necessary skillset to be successful in the SE work environment. While other researchers have tried to help students with planning, existing interventions often supported an individual student rather than a team and typically did not promote iterative reflection and feedback on a collaboratively created plan.

In our research, we propose Software Engineering Team Strategic Planning (SETS Planning), a novel strategic planning and reflection technique that helps students plan the underlying development tasks and reflect on their outcome. We scaffold planing and reflection on the plan to encourage Self-Regulated Learning (SRL) among students. The planning process helps teams identify underlying development tasks and divide them into smaller manageable sub-tasks. The reflection helps each individual visualize their performance and make a more accurate plan in the next milestone. We introduced this new approach of planning and reflection to undergraduate SE students in Fall 2021 and Spring 2022 semesters. We investigated the following research questions:

RQ1: How helpful/difficult do students find it to plan their tasks using the SETS Planning technique? Why do they find it helpful/unhelpful?

RQ2: To what extent do student task outcomes deviate from their plans? What patterns can be identified from the deviation?

RQ3: To what extent do actual sub-task completion times differ from planned times estimation? Are there tendencies toward under-/over-estimation?

RQ4: What, if any, patterns between teamwork and task performance can be identified?

Chapter 2

Literature Review

SE students need self-regulated learning skills to ’become masters of their own learning’ [Zimmerman(1990)] and effectively contribute to a group development project. A typical SRL model usually involves three components: meta-cognition, strategic planning, and self-motivation [Chung and Hsiao(2020)]. Metacognition is the awareness and understanding of one’s own thought processes. It helps to understand why and when to use a particular strategy. Strategic planning includes dividing work among team members, scheduling work according to deadlines, and estimating task completion time. Self-motivation is the confidence of an individual that he or she can perform a specific task or can achieve the goal. In this section, we review prior work on strategies used to promote SRL in SE education.

2.1 Strategies for Metacognition in SE

Researchers have tried different strategies to encourage metacognition among SE students. Some researchers tried to monitor the consistency of use of self assessment tools to assess the metacognition ability of students [Chung and Hsiao(2020)], [Bergin et al.(2005)], [Martin et al.(2015)]. Personal schedule sheets have been used to reduce procrastination [Martin et al.(2015)]. Self-monitor time spent with different activities and reflect about their actions through visualization is another strategy [Meyer et al.(2017)]. Others interview [Kazerouni et al.(2017)] or use grades of assignments to assess the metacognition ability of students [Edwards et al.(2009)].

Most of the prior work that supports metacognition involve students making individual plans or schedules, then measuring adherence to the plan through self assessment or a monitoring tool. This line of work does not consider the teamwork and coordination required for software engineering team development.

2.2 Strategies for Strategic Planning in SE

Researchers have also tried implementing strategies for strategic planning to motivate SRL. One approach is to help teams schedule meetings and make agendas and action items [Gamble and Hale(2013)], [Geyer et al.(2001)] and track student involvement during meetings [Gamble and Hale(2013)]. Another approach for planning is to use a tool to divide and track tasks among teammates from user stories [Gamble and Hale(2013)], [Geyer et al.(2001)]. Tagging related study items with each task can facilitate team collaboration [Jorgenson et al.(2011)]. Other researchers considered Peer feedback to find the active participation in the project development [Meyer et al.(2019)].

Prior strategies to Support Strategic Planning including tools to help teams schedule meetings, help teams create and assign tasks from user stories, and measuring adherence to a process like Agile. This line of work typically assesses student planning only after the project is complete, rather than explicitly promoting iterative reflection and feedback on a collaboratively created plan. Planning and reflection are often separate, unsupported tasks.

2.3 Strategies for Motivation in SE

To encourage the motivation of students for SRL researchers used techniques like sending follow up emails [Martin et al.(2015)], monitoring the coding behaviours by looking at number of erroneous code, number of successful or failed test run, code result

[Kazerouni et al.(2017)], watching the log time frequency of study material [Chung and Hsiao(2020)], and detecting procrastination [Martin et al.(2015)], [Kazerouni et al.(2017)]. Helping the student to be motivated can stimulate SRL. However, we decided to focus on strategic planning, and metacognition about the planning process, for this work.

Chapter 3

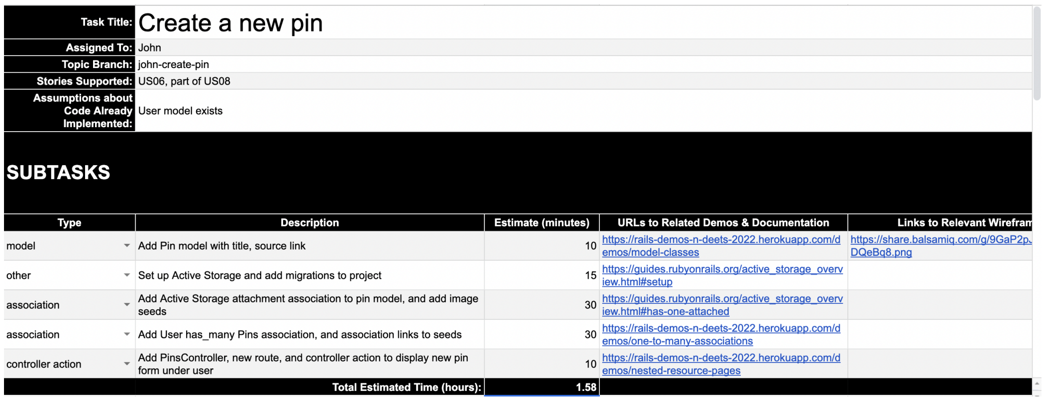
Software Engineering Team Strategic Planning (SETS Planning)

The SETS Planning has two parts- Planning and Reflection. Students were introduced to the SETS Planning technique at the start of their course project, including examples and demo videos for how to use the spreadsheet. Planning and Reflection are completed for each development milestone of the project. We used Google Spreadsheets to implement the SETS Planning technique for our pilot test. Each team has a single google Sheet. Each tab of the sheet is for planning a single development task. We provided a planning and reflection template for students to fill out as they work with their team.

3.1 Planning

Before engaging with the SETS Planning Sheet (Figure: 1), the team determines which development tasks belong in this milestone and assign each task to a team member.

First, the team fills in meta data about the task, like a title, the topic branch name, which team member is responsible for this task, and which user stories this tasks supports. Second, students articulate their assumptions about what code should already be implemented before this task begins. Third, they work as a team to identify all the



Task Title:	Create a new pin
Assigned To:	John
Topic Branch:	john-create-pin
Stories Supported:	US06, part of US08
Assumptions about Code Already Implemented:	User model exists

SUBTASKS

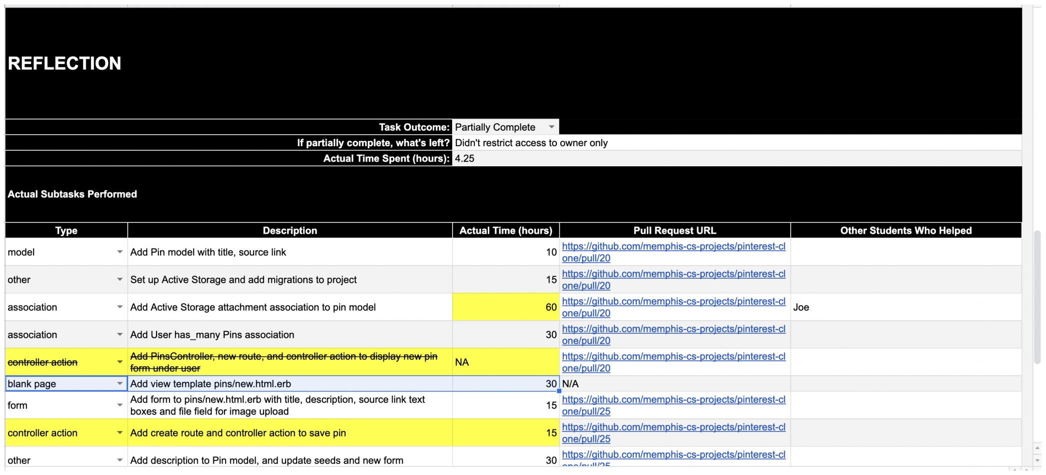
Type	Description	Estimate (minutes)	URLs to Related Demos & Documentation	Links to Relevant Wirefram
model	Add Pin model with title, source link	10	https://rails-demos-n-deets-2022.herokuapp.com/demos/model-classes	https://share.balsamiq.com/g/9GaP2pJ DQeBq8.png
other	Set up Active Storage and add migrations to project	15	https://guides.rubyonrails.org/active_storage_overview.html#setup	
association	Add Active Storage attachment association to pin model, and add image seeds	30	https://guides.rubyonrails.org/active_storage_overview.html#has-one-attached	
association	Add User has_many Pins association, and association links to seeds	30	https://rails-demos-n-deets-2022.herokuapp.com/demos/one-to-many-associations	
controller action	Add PinsController, new route, and controller action to display new pin form under user	10	https://rails-demos-n-deets-2022.herokuapp.com/demos/nested-resource-pages	
	Total Estimated Time (hours):	1.58		

Figure 1: Planning process of SETS Planning spreadsheet

sub-tasks needed to implement the task and tag the appropriate sub-task type (e.g. model, controller, etc.). Sub-tasks are the actual coding steps needed to perform in order to implement the task. Fourth, the developer assigned to this task estimates their time to complete each sub-task. From the estimates, total time to complete a task is calculated automatically. Finally, they add links to class resources and wireframe for associated sub-tasks.

3.2 Reflection

We encouraged students to fill out the reflection section of the template as they were coding (Figure: 2), although many waited until they had finished the task or until the milestone deadline. First, students copy the sub-task list and estimated times from the plan to the reflection. Second, students adjust the plan to reflect what actually happened. They manually highlight any changes in the list of sub-tasks (something added, deleted) and if the actual development time differed from their estimate. The Actual time to complete the task is automatically calculated from the actual development time for each sub-task. Third, students provide the GitHub pull request links and list if any teammate helped to implement a sub-task. Finally, we asked students to give explanations for significant deviations from the Plan/Estimates in a separate column. Students reported the task



REFLECTION

Task Outcome:	Partially Complete
If partially complete, what's left?	Didn't restrict access to owner only
Actual Time Spent (hours):	4.25

Actual Subtasks Performed

Type	Description	Actual Time (hours)	Pull Request URL	Other Students Who Helped
model	Add Pin model with title, source link	10	https://github.com/memphis-cs-projects/pinterest-clone/pull/20	
other	Set up Active Storage and add migrations to project	15	https://github.com/memphis-cs-projects/pinterest-clone/pull/20	
association	Add Active Storage attachment association to pin model	60	https://github.com/memphis-cs-projects/pinterest-clone/pull/20	Joe
association	Add User has_many Pins association	30	https://github.com/memphis-cs-projects/pinterest-clone/pull/20	
~~controller action~~	~~Add PinsController, new route, and controller action to display new pin form under user~~	NA	https://github.com/memphis-cs-projects/pinterest-clone/pull/20	
blank page	Add view template pins/new.html.erb	30	N/A	
form	Add form to pins/new.html.erb with title, description, source link text boxes and file field for image upload	15	https://github.com/memphis-cs-projects/pinterest-clone/pull/25	
controller action	Add create route and controller action to save pin	15	https://github.com/memphis-cs-projects/pinterest-clone/pull/25	
other	Add description to Pin model, and update seeds and new form	30	https://github.com/memphis-cs-projects/pinterest-cl	

Figure 2: Self-report Reflection spreadsheet of SETS Planning

outcome: complete, partially complete or not started. If partially complete, students describe what parts are remaining.

Chapter 4

Method

We conducted a pilot test over Fall 2021 and Spring 2022 semester in an undergraduate Software Engineering class to evaluate the impact of the SETS Planning technique.

4.1 Participants

This experiment involved 75 undergraduate computer science students (43 and 32 from Fall and Spring semesters respectively). Students from Fall semester were divided into 11 teams of 2 or 4 individuals. Students from Spring semester were divided into 9 teams of 3 or 4 individuals. Both semesters were taught by the same instructor.

4.2 Classroom Setting

In the first half of the semester, students attend lecture and complete skills assignments to learn Ruby on Rails. In the second half of the semester, students complete a software project with a team. The time for the software project is divided into three milestones: M0, M1 and M2. During M0, teams were assigned a project and asked to complete user stories, definitions, wireframes, and database design. M1 and M2 were dedicated for planning, development, and reflection.

4.3 Procedure

Students were instructed to complete their plan for the milestone at the start of M1 and M2, and fill out the reflection as they were developing a sub-task. Students were asked to work together to create the plan and distribute tasks equally among teammates. Each teammate is required to work a minimum of 10 hours in each milestones. At the end of each semester, we collected student opinions of the process with a Student Feedback Survey. The survey inquires the student opinions about planning, reflection, team collaboration, and usefulness of SETS Planning. At the end of the Fall semester, we conducted an interview with the course teaching assistant (TA) to gather her opinions about the impact of this intervention on grading.

In the Fall semester, students were not asked to do any planning during the first half of the semester (lecture and individual Skills Assignments). In the Spring semester, we introduced the SETS planning process during the Skills Assignments at the start of the semester to give students more time to practice planning on their own before the team project.

4.4 Data Collection and Analysis

We collected student survey responses, student-created plans and reflections from development milestones, and TA interview data. In total there are 52 survey responses (31 and 21 for Fall and Spring respectively).

We collected SETS plans and reflections from all teams for all milestones. However, some students did not fully participate in the SETS process. We deem these students had 'no effort reflection' for that specific task and removed their data from analysis.

There are 2 ways to be deemed 'no effort': 1) if a task has a blank reflection, or 2) the sub-task description AND actual time that is exactly copied from the plan. If a task was

Table 1: Total removed and included tasks based on No Effort Reflection

Semester	Milestone	Data Removed	Data Included
Fall	M1(n=117)	19	98
Fall	M2 (n=105)	12	93
Spring	M1 (n=74)	16	58
Spring	M2 (n=64)	17	47

moved to next milestone, we also removed the data from the current milestone. See table 1.

After collecting data we replaced any identifier like student name, team name, and GitHub username with unique anonymized identifier. For example, student names were replaced by unique participant ID (e.g. P1-F, P2-S) where F and S represents Fall and Spring semester respectively. As our research involves human subjects, we obtained IRB approval for this project (IRB Number: PRO-FY2022-287, Approval date: Jan 24, 2022). Our IRB determined this was exempt research.

We analyze data from three perspectives: student perceptions of the experience, how students used and deviated from their plans, and teamwork challenges.

To analyze the student plans, we counted the number of task assigned to a teammate in each milestone and compared across teammates and across teams. If a team did not have an even task distribution, the individual assigned most tasks is denoted as high performer and an individual assigned least task is denoted as low performer.

To analyze the student reflections, we calculated the average number of sub-tasks that are added or deleted per task and the number of sub-tasks for which the actual time changed. To analyze students' improvement with estimating time for development tasks, we calculated the time deviation for each task. Time deviation is calculated by subtracting the Total Estimation from the Actual time.

We interviewed the TA of SE course in the Fall semester that was conducted over Zoom and was video recorded. We follow a semi-structured protocol asking for TA satisfaction with student plan, TA perception of why student struggled to make good plan, and how SETS Planning can help TAs provide help and meaningful feedback to student. The interview was approximately 20 minutes long.

Chapter 5

Result

Recall that we analyzed data from three perspectives: Student perception, Plan deviation, and Teamwork challenges. We also report the Teaching Assistant's perception of the SETS Planning technique.

5.1 Student Perception of Helpfulness and Difficulty

Students had generally positive responses to SETS Planning. 74% students form Fall and 64% students from Spring semester agreed that the task plan was helpful. Students in the spring had a slightly more positive experience, perhaps due to the instructor's familiarity with the method and the decision to introduce SETS Planning earlier in the semester. See figure 3 for responses to all Likert scale questions.

5.1.1 This process helped me identify and organize task lists

Students replied that the task plan served as a checklist of tasks. They can refer back to the plan while coding and check what exactly needed to develop next. Participants (P8-F, P20-F, P24-F, P35-F, P42-F, P6-S, P14-S, P23-S, P24-S, and P31-S) felt that the task plan served as a checklist of manageable task that they followed while developing . P35-F said "It gave a general guideline on what I needed to do, so I could use those to get an idea on

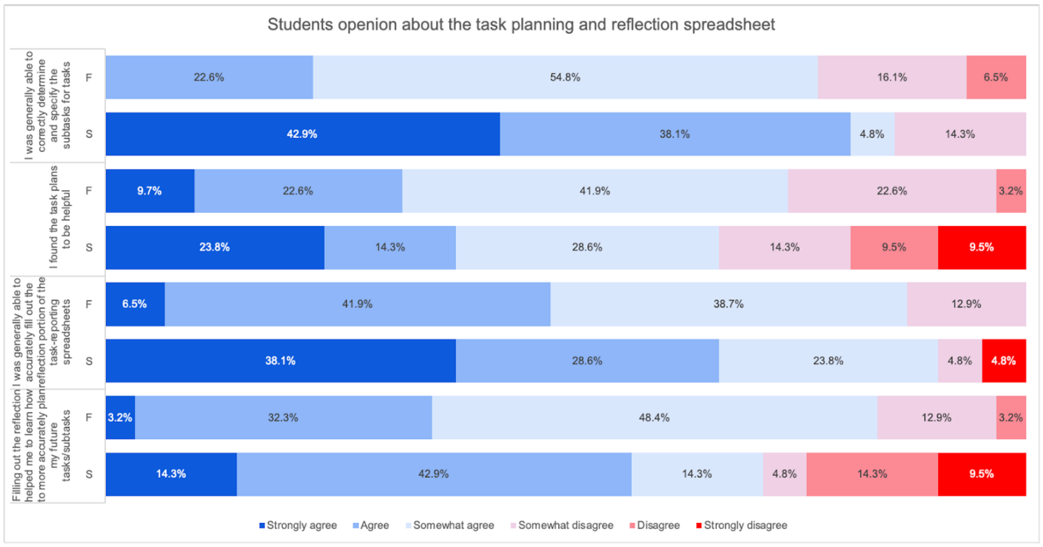


Figure 3: Students' responses to Likert scale questions

the next best step" and P3-S stated "The task plans were helpful in keeping track of what I should do next and making sure I didn't skip small steps...".

10 out of 31 participants from Fall and 4 out of 21 participants from Spring mentioned that the task plan helped them to organize task. P36-S said that "It helped me to organize what I needed to do into smaller, easier to digest chunks, therefore I felt less stressed about completing a task". P23-S said "I feel like the task plans do help keep organized, and make sure nothing is missed...".

5.1.2 How the reflection helped

87% from Fall and 91% from Spring agreed that they were able to accurately fill out the reflection (Figure 3). P11-F said "The reflection was kind of easy. I say that because if you are able to do them while you are trying to complete your task then you can accurately account for the time and the sub-task. But, if you wait until after then you might miss something and not have a good reflection".

84% student from Fall and 72% student from Spring agreed that the reflection helped them to more accurately plan the future sub-task (Figure 3). P8-F said "It made me see what features needed to be improved on or if I did not finish something" and P8-S stated

"I gained experience through my completion of the reflections in the past so I knew what possible pitfalls could occur during the execution and how to document them". P34-F, P40-F and P2-S felt that they learned to breakdown a task and that was useful for future planning. P40-F said " I learned to break down my work a little bit more that may make it more approachable for some to understand".

Reflection helped students to estimate better in the next iteration. P26-F said "The reflections made me reconsider the amount of time I'd need to allot to the next planned tasks". P25-S said "I would say finding out how long it took me to do tasks in the first task plan helped me to understand how long the tasks would take in subsequent task-plans".

According to some students, filling out the reflection is easy when they report it as they were implementing a sub-task. P41-F said "It helped me to learn that I need to fill out the reflection as I go instead of all at once in the end".

5.1.3 Difficulties in planning

Approximately 77% from Fall and 56% from Spring agreed that they were able to identify sub-tasks for a task. On the other hand, 23% participants form Fall and 14% participants from Spring disagreed to the statement (Figure 3). Students reported a variety of reasons why they struggled with sub tasks. P39-F, P4-S, and P7-S said that they did not understand what sub-tasks are important to put on to the plan. P4-S stated "I had trouble filling out the plan section because sometimes I did not know what exactly I need to put into it and what was really important to put in it ". P17-F, P32-F, and P11-S said they did not have enough experience. P18-F and P11-S reported trouble in teamwork coordination because tasks are interconnected and specifying sub-tasks for overlapping tasks are problematic. P11-S said "Another problem was team coordination; my team's organization/coordination was not very good, so if someone was already working on something that I needed for one of my tasks and they were doing it differently than how I planned to do it, I had to adapt to the way they were doing it because I didn't want to break the project by changing the code."

Participants (P8-F, P20-F, P24-F) found it difficult to fulfill the minimum 10 hour of work per teammate and for other participants (P20-F, P5-S, P13-S, P32-S) the restriction of maximum 40 minutes estimate per sub-task was difficult. P8-S said "I think matching to the 10hrs of work was difficult. My team focused a lot on the time restraints instead of the sub-tasks". P32-S said "... you can't put more than 40 minutes in a project task plan worksheet. I understand why it was added, but it felt constraining since there's only so many nuances that I can or want to go into when coming up with a task plan."

5.1.4 Reporting actual time in the reflection was hard to do accurately

Only 26% student from Fall and 33% student from Spring semester reported that the reflection portion of SETS Planning was unhelpful (Figure 3). Participants (P5-F, P21-F, P22-F, P43-F, P4-S, P7-S, P10-S) did not found the reflection helpful. P4-S said "I honestly did not use much of the reflections portion after I filled it out because I just talked with my teammates or moved onto the other tasks I had to do instead of looking back to the reflections ". Many of the participants who did not find the reflection helpful are those were 'no effort' reflectors.

For some students, filling out the reflection was difficult because they did not keep track of time while working. 10 out of 31 participants from Fall and 3 out of 21 participants from Spring reported they did not keep track of time while coding. P32-S stated "I don't keep track of how long it takes me to do things as it's too much of a hassle to worry about when I'm already focused on the task at hand. So consequently I end up guessing a lot on how long it took me to do things."

We found that fewer students from Spring faced difficulties than students from Fall. We assume that because we introduced the planning technique earlier during Skills

Table 2: Mean and Standard Deviation of sub-task added and deleted per task in each milestone for Fall and Spring semesters

Semester	Milestone	Subtask Added		Subtask Deleted	
		Mean	SD	Mean	SD
Fall	M1 (n=98)	3.58	4.78	1.47	1.83
Fall	M2 (n=93)	3.03	3.92	2.3	3.28
Spring	M1 (n=58)	3.59	5.81	2.62	4.53
Spring	M2 (n=47)	4.13	7.74	1.09	2.24

Assignment to the students in the Spring term, they were more familiar with the planning process and prepared to do the planning for the project.

5.2 Plan Deviation

In the reflection, students reported sub-task that they added or deleted and actual time they took to finish a sub-task. Most of the sub-tasks students added or deleted are related to Ruby MVC (model, view, controller) and bug-fixing. Table 2 shows the mean and SD of sub-task added and deleted per task.

5.2.1 I don't have the knowledge to make a perfect plan

Students often deviate from their plan because they do not have enough knowledge (P17-F, P21-F, P36-F, P40-F). P40-F said "The level of detail for subtasks can be hard to pin down, and sometimes a lack of knowledge can make it even more difficult as we are still learning/strengthening knowledge a great deal as we go". For some students, the planning process took much effort (P5-F, P11-F, P28-F, P32-F, P33-F, P34-F, P-39, P43-F, P4-S, P6-S, P17-S, P24-S, P32-S). P32-S said "They are good for laying out objectives, however the catch is that I found them a bit of a hassle to fill out."

5.2.2 I stopped using the plan when the plan changed

6 out of 31 students from Fall and 3 out of 21 students form Spring term reported that they did not follow the plan while developing. Others stopped following the plan when

they discovered new sub-tasks while coding. P11-S said "... when I actually started coding, I realized there were many more things needed to complete my task that I didn't plan". P38-F said "...planning things before coding does not capture everything that really needs to be done. Most coding tasks I've realized while in the middle of a task I had planned out already".

Estimating the time for a sub-task is difficult specially for the novice programmers. Students often tend to over estimate time. Students' time estimation error was usually within -2 to +2 hours. The SD of time error in M1 and M2 are 1.86 and 1.67 respectively for Fall and SD of time error in M1 and M2 for Spring are 1.66 and 2.89 respectively. Some tasks has time error more than +4 hours and only 2 tasks are over +10 hours. Unexpected issues were usually due to the complexity of Ruby coding format, setting database migration, and bug-fixing. P6-S said "Comment controller and interaction with other models was a lot more complicated than expected". Team-2-S reported "it took too much time to solve a problem because of teamwork challenges" (P4-S).

The tendencies toward overestimate is higher in M1 than M2 in both semesters. Students tend to over-estimate not only because they don't know how much time they should allocate but sometimes they think over-estimate is better than under-estimate (P36-F, P12-F, and P10-S). P36-F said "The reflections caused me to evaluate how I planned my tasks in the future. Especially time wise and how it was better to overestimate needed time than to underestimate it".

5.3 Teamwork Challenges

One of the major teamwork challenge is not distributing tasks equally among teammates. Though we asked teams to distribute work evenly among teammates, but most of the teams did not follow the instruction. There were only two fully balanced teams (Team-6-F and Team-2-S) who assigned equal tasks among teammates in both M1 and M2 milestones. Other teams Team-5-F, Team-9-F, and Team-8-S are balanced in M1 but

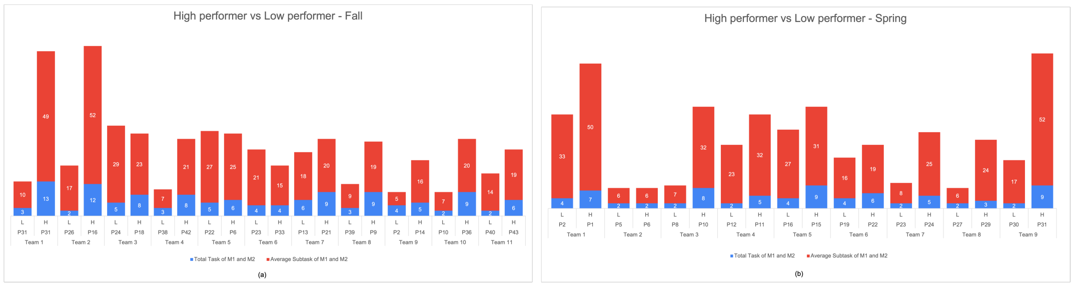


Figure 4: High and Low performers' contribution graph within teams for Fall (a) and Spring Semesters (b).

not in M2. Rest of the teams are totally unbalanced (tasks are unevenly distributed among teams in each milestones). At least one individual of unbalanced teams is assigned most tasks. We call the person High performer, and the person with lowest number of tasks is called Low performer. The High performer also come up with most number of sub-tasks. Figure 4 shows how the SETS Plan technique also helps reveal which teams are unbalanced and which students are high or low contributors.

5.3.1 My team needs to communicate to be successful

We surveyed students about their team's collaboration. We found that each teammate of Team-2-F, Team-3-F, Team-8-F, and Team-7-S reported no teamwork related challenges. P5-F said "... we were pretty on top of planning what each person should do". Other teams had mixed opinions.

Two main reasons for team conflict are: interconnected tasks (P8-F, P27-F, P35-F, P42-F, P14-S) and poor communication (P3-F, P21-F, P32-F, P36-F, P4-S, P7-S, P10-S, P11-S, P13-S). P38-F said "Most discrepancies from my task reporting was relying on waterfall assignments. For example, it was reliant on other sections/subsection to be implemented before I could really create any significant progress towards my tasks". P21-F said "... The group meetings in person never really happened after M0 and barely on discord. So, I had to figure out what I need to do and what he need to do and what he have done if he doesn't tell me."

5.4 TA perception of SETS Planning

The TA felt that SETS planning is helpful because she can visualize the whole project at a glance. If students are able to break down steps, connect them to wireframes, and estimate completion times, the TA can provide meaningful feedback and better help to students. After helping a student to prepare the plan, the TA said “In the beginning, he had no idea how to do his tasks because he was far behind in the demos, so I helped him. Eventually, he learned and ended up with a good quality plan”. We shared high quality and low quality plans with the TA and asked her “If students get stuck while coding and show you this plan, would you be able to help them?” The TA replied “Yes, with this level of detail I would definitely identify what problems they are going to have and [where] they might get stuck”.

When asked about challenges with planning, the TA identified several areas of struggle. Helping students can be challenging because students do not have enough knowledge about Ruby coding. Sometimes helping student teams is difficult because they did not made the plan together. Students were told to make the plan together and report reflection individually, but some of the teams did not follow the guideline. The TA said “Students did not make the plan together. That’s why their initial plans were the wrong type of things”.

Chapter 6

Discussion & Future Work

Our experiment suggests that SETS Planning is beneficial to SE student teams. SETS Planning can help students to identify and plan the underlying development tasks. It also helps teams to keep tasks organized and avoid major project implementation failure. Teams can get better idea of how much time they should invest in each development steps. Scaffolding the planning and reflection can help teams to identify where to focus in next iteration. In other words, student teams can learn to self-regulate their learning.

SETS Planning helps to reveal high and low contributors within teams. This information can help instructor while grading software project. TAs can also provide meaningful feedback by looking at student plans.

There are several limitations to our experiment. First, we did not assess student plans nor did we provide feedback to their initial plans. Assessing and providing feedback to student plans would have required additional course staff that was not available for the pilot test. Second, tracking the changes in the reflection is difficult for students. Future work might consider building a task-tracking system to help streamline the process, particularly the time tracking aspect of reflection. Third, students were asked to create the plan as a team, but some teams did not follow this instruction. These teams also had other instances of team conflicts. Future work could explore how to motivate struggling teams to commit to a planning process and examine how this process impacts struggling teams differently than harmonious teams. Despite these limitations, the data generated by this pilot test provides useful insights for educators and researchers who work to teach SE student teams to self-regulate their learning.

Chapter 7

Conclusion

SETS Planning is a technique to help SE student teams make a plan for project development and reflect on collaborative planning. This technique helps teams keep development tasks organized and easy to follow. The reflection helps students improve their planning skills for the future milestones. Scaffolded planning and reflection strategy of SETS Planning can help SE student teams to self-regulate their learning. Instructors and TAs can also benefit from this data, as it helps identify the high and low performer within a group for grading and provide additional context when giving feedback to students.

References

[Bergin et al.(2005)] Susan Bergin, Ronan Reilly, and Desmond Traynor. 2005. Examining the role of self-regulated learning on introductory programming performance. In *Proceedings of the first international workshop on Computing education research*. 81–86.

[Chung and Hsiao(2020)] Cheng-Yu Chung and I-Han Hsiao. 2020. Investigating patterns of study persistence on self-assessment platform of programming problem-solving. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 162–168.

[Edwards et al.(2009)] Stephen H Edwards, Jason Snyder, Manuel A Pérez-Quiñones, Anthony Allevato, Dongkwan Kim, and Betsy Tretola. 2009. Comparing effective and ineffective behaviors of student programmers. In *Proceedings of the fifth international workshop on Computing education research workshop*. 3–14.

[Gamble and Hale(2013)] Rose F Gamble and Matthew L Hale. 2013. Assessing individual performance in Agile undergraduate software engineering teams. In *2013 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1678–1684.

[Geyer et al.(2001)] Werner Geyer, Heather Richter, Ludwin Fuchs, Tom Frauenhofer, Shahrokh Daijavad, and Steven Poltrock. 2001. A team collaboration space supporting capture and access of virtual meetings. In *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*. 188–196.

[Jorgenson et al.(2011)] Noah M Jorgenson, Matthew L Hale, and Rose F Gamble. 2011. SEREBRO: facilitating student project team collaboration. In *Proceedings of the 33rd International Conference on Software Engineering*. 1019–1021.

[Kazerouni et al.(2017)] Ayaan M Kazerouni, Stephen H Edwards, T Simin Hall, and Clifford A Shaffer. 2017. DevEventTracker: Tracking development events to assess incremental development and procrastination. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. 104–109.

[Martin et al.(2015)] Joshua Martin, Stephen H Edwards, and Clfford A Shaffer. 2015. The effects of procrastination interventions on programming project success. In

Proceedings of the eleventh annual International Conference on International Computing Education Research. 3–11.

[Meyer et al.(2017)] Andre N Meyer, Gail C Murphy, Thomas Zimmermann, and Thomas Fritz. 2017. Design recommendations for self-monitoring in the workplace: Studies in software development. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (2017), 1–24.

[Meyer et al.(2019)] André N Meyer, Gail C Murphy, Thomas Zimmermann, and Thomas Fritz. 2019. Enabling good work habits in software developers through reflective goal-setting. *IEEE Transactions on Software Engineering* 47, 9 (2019), 1872–1885.

[Zimmerman(1990)] Barry J Zimmerman. 1990. Self-regulated learning and academic achievement: An overview. *Educational psychologist* 25, 1 (1990), 3–17.