Investigating Reflection in Undergraduate Software Development Teams: An Analysis of Online Chat Transcripts

Christopher Hundhausen
chris.hundhausen@oregonstate.edu
Oregon State University
Corvallis, Oregon, USA

Phill Conrad
phtcon@ucsb.edu
UC Santa Barbara
Santa Barbara, California, USA

Olusola Adesope
olusola.adesope@wsu.edu
Washington State University
Pullman, Washington, USA

Ahsun Tariq
tariqa@oregonstate.edu
Oregon State University, Corvallis
Oregon, USA

Samir Sbai
samir.sbai@wsu.edu
Washington State University, Pullman
Washington, USA

Andrew Lu
alu@ucsb.edu
UC Santa Barbara, Santa Barbara
California, USA

ABSTRACT

Metacognition is widely acknowledged as a key soft skill in collaborative software development. The ability to plan, monitor, and reflect on cognitive and team processes is crucial to the efficient and effective functioning of a software team. To explore students' use of *reflection*—one aspect of metacognition—in undergraduate team software projects, we analyzed the online chat channels of teams participating in agile software development projects in two undergraduate courses that took place exclusively online (n = 23 teams, 117 students, and 4,915 chat messages). Teams' online chats were dominated by discussions of work completed and to be done; just two percent of all chat messages showed evidence of reflection. A follow-up analysis of chat vignettes centered around reflection messages (n = 63) indicates that three-fourths of the those messages were prompted by a course requirement; just 14% arose organically within the context of teams' ongoing project work. Based on our findings, we identify opportunities for computing educators to increase, through pedagogical and technological interventions, teams' use of reflection in team software projects.

CCS CONCEPTS

• **Software and its engineering → Programming teams**; • **Social and professional topics → Software engineering education**; • **Human-centered computing → Computer supported cooperative work**.

KEYWORDS

team software projects, software engineering, software engineering education, metacognition, reflection, content analysis, agile

ACM Reference Format:
Christopher Hundhausen, Phill Conrad, Olusola Adesope, Ahsun Tariq, Samir Sbai, and Andrew Lu. 2023. Investigating Reflection in Undergraduate Software Development Teams: An Analysis of Online Chat Transcripts. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569790

1 INTRODUCTION

Undergraduate computing students benefit greatly from team software development projects; these are crucial to enabling them to develop the skills needed to succeed in the software profession. While undergraduate computing students typically acquire adequate training in technical (coding) skills, studies of professional software development suggest that they may possess deficiencies in soft skills (e.g., communication, collaboration) when they enter new jobs in the profession [5, 14, 42]. This suggests the need to develop more effective pedagogical approaches for team software development projects that emphasize training in soft skills.

Metacognition is widely acknowledged as a key soft skill in collaborative software development [2, 38, 45]. Metacognition encompasses the ability to plan, monitor, and reflect on cognitive and team processes. It is crucial to the effective functioning of a software team [43]. It enables a team to adapt not only to the changing requirements and ill-structured nature of software development projects, but also to the strengths and interests of team members.

For software teams, *reflection*—assessing the team's process to determine what is working, what is not working, and what changes could be made—is a particularly important aspect of metacognition. The Agile development approach is emphatic about the need for software teams to reflect regularly on their process, and prescribes the *retrospective* as a structured way for teams to do this [4].

The importance of reflection in team software development—and the need to develop effective pedagogical approaches to promote reflection—leads to the following research question:

RQ: To what extent do student software teams demonstrate reflection as part of their process?

We present an exploratory analysis of the use of reflection in the online chats of undergraduate teams engaged in software development projects. We focus on the required team projects of two undergraduate courses conducted completely online during the pandemic at two large North American research universities (n = 23 teams, 117 students). To explore our research question, we performed a content analysis of the students teams' chat channels

(n = 4,915 messages). To gain further insight into how and why reflection emerged in these chats, we conducted a follow-up analysis of the chat vignettes in which reflection messages were embedded.

Our results indicate that explicit reflection was a rarity in teams' online chats: *on average, just two percent of teams' chat messages showed evidence of reflection.* Moreover, 76% of the chat messages with reflective content were prompted by *course requirements*; only 14% arose organically through teams' ongoing software development activities. We consider the implications of these results for the design of pedagogical and technological interventions to encourage greater use of reflection in team software development projects.

2 RELATED WORK

2.1 Soft Skills in Computing

Soft skills (e.g. communication, collaboration, teamwork) are widely acknowledged to be essential in the software profession [29]. Yet, numerous studies have shown a deficit of these skills in graduates of undergraduate computing programs [5, 6, 12, 14, 37, 42].

Given their importance, computing educators and researchers have explored many approaches to teaching soft skills (see [25] for a review). While some approaches target specific soft skills such as creativity [35], communication [8], or teamwork [23], others aim for more comprehensive training in a variety of soft skills [22, 27]. Likewise, while some soft skills training has been embedded in capstone design courses [1, 13] educators have explored soft skills training in other venues, including courses in game design [9] and even entire courses dedicated to soft skills [27].

Soft skills prove particularly challenging to evaluate due to their subjective nature. Computing educators have traditionally relied on some combination of peer evaluations [33] and instructor assessment. There have also been systematic efforts to develop structured soft skills metrics and rubrics [10, 18, 28, 41]. Our work contributes to broader efforts to integrate soft skills training and evaluation into computing education by studying students use of a particular soft skill (reflection) in course activities and considering ways that students' use of the skill might be prompted.

2.2 Reflection in Software Development

Reflection refers to the practice of assessing one's processes, behaviors, and progress relative to goals, with an eye toward identifying improvements to be made. Schön [44] distinguishes between two key forms of reflection: *reflection-in-action*, in which one assesses one's practices while in the midst of performing them; and *reflection-on-action*, in which one assesses one's practices after the fact. Our analysis focuses on the latter.

Reflection is widely recognized as a hallmark of professional design and engineering practice [44]. Professional software engineers have been interested in adopting reflective practices [20], while researchers have explored tools and techniques to promote reflection in the software engineering process (e.g., [7]). In computing education, researchers have performed detailed investigations of the use of reflection in student software development (e.g., [19]) and explored pedagogical approaches to promote reflection, with a particular emphasis on studio-based learning [11, 26]. Our work contributes to this line of work by investigating the use of reflection in student software teams' chat channels and considering ways to promote reflective practice in undergraduate team projects.

Reflection is a key component of the more general practice of *metacognition*, which encompasses planning, monitoring, and reflecting on thinking and behavior [36]. Widely studied in computing education [34, 39], metacognition has been shown to be a crucial soft skill in professional software development [2, 38, 45]. Moreover, it has been identified as being particularly deficient in studies of new software developers in industry [5, 6]. By studying student teams' use of reflection and how it arises in the software development process, our work aims to contribute to addressing this deficiency through improved pedagogical and technological interventions.

2.3 Studying Software Team Communication

This study performs a content analysis [31] of the online chat communications of virtual teams engaged in software development projects. In the field of software engineering, a large body of research has studied software team communication; see [17] for a comprehensive review. A few of these studies have, like us, performed content analyses of team chats to better understand team communication and find evidence of best practices (e.g., [3]). However, ours is the first study, to our knowledge, to investigate software team chat communications for evidence of reflection.

In a similar vein, the field of computer-supported collaborative learning (CSCL) is interested in understanding collaborative learning through analyses of group communications [40]. To analyze communication in our study, we draw on an influential theoretical framework developed through this work: *social interdependence theory* [16], which holds that positive interdependence in collaborative learning is characterized by the following behaviors:

- help-giving and help-seeking
- giving and receiving feedback
- challenging and encouraging each other
- jointly reflecting on progress and process

Based on this theoretical framework, Curtis and Lawson [15] created a content coding scheme to analyze chat transcripts in online collaborative learning. Swigger et al. [46] demonstrated the value of this coding scheme in analyzing software team communication. Given the presence of reflection in the framework and its demonstrated utility in studies of software teams, we adopted it as a foundation for the coding scheme used in this study.

3 METHODS

3.1 Courses and Participants

This study occurs in the context of two courses offered completely online during the Covid-19 pandemic in academic year 20-21:

Course A ("Advanced Application Development") was taught by the second author at UC Santa Barbara, a large North American research university. It is a 10 week course taken primarily by second and third year undergraduate computer science majors. It focuses on the development of full-stack web applications through a team project lasting the duration of the term. The course emphasizes both technical skills and soft skills related to Agile practices (standups, sprints, Kanban, user stories, acceptance criteria) and GitHub workflows (pull requests, code reviews).

Course B ("Web Development") was taught by the first author at Washington State University. It is a 15 week advanced undergraduate course in full-stack web development. During the first 10 weeks, students learn web programming through lectures, live coding demos, and a series of individual assignments that build upon each other to produce a full-stack web app. In the final five weeks, students form teams and apply what they have learned to build a full stack web app of their choice. As part of the team project curriculum, students learn about the same Agile practices and GitHub workflows emphasized in Course A.

Both courses took place during the pandemic and were conducted *completely online*. Class meetings were held synchronously through Zoom. Outside of class meetings, students engaged with online learning materials and communicated via online communication tools both synchronously and asynchronously.

Table 1 presents demographic data on study participants. The study involved three separate offerings of Course A and one offering of Course B. The study was approved by the Institutional Review Boards of each university; students could opt in to the study by signing an informed consent form. For a team's chat data to be included in the study, all team members had to consent. As shown in Table 1, in the four courses involved in this study, all members of 23 of the 46 teams (*n*=117 students) consented to participate.

Table 1: Study Demographics

Course	Teams Enrolled	Students Enrolled	Teams Consenting	Students Consenting				
				n	M	F	U	Mean Age
A1	12	66	7	38	30	8	0	19.9
A2	10	54	2	9	9	0	0	20.7
A3	12	64	2	11	8	3	0	19.7
B	12	59	12	59	50	4	5	23.3
Total	**46**	**243**	**23**	**117**	**97**	**15**	**5**	**20.05**

3.2 Materials: Team Project

Table 2 presents information about the team software development projects in each course. In Course A, teams of up to six members were formed through a CATME team-building survey [32] and then randomly assigned to a legacy code project that had been developed by students in previous course offerings. In contrast, students in Course B were given the option of proposing their own web development project or choosing from a list of pre-approved projects. Students were asked to form their own teams of three to five students; students who did not sign up for a team were randomly assigned to teams that had openings available In both courses, student teams were given starter code—either the code base from a legacy code project (Course A), or a code base from a demo project developed in the first part of the course (Course B).

Teams in both courses undertook projects of three to five weeks, split up into three to four sprint cycles. Students used GitHub as their source code repository, along with GitHub issues and project boards to monitor and track their development activities. In addition, teams used channels in either Slack (Course A) or Microsoft Teams (Course B) for online chat communications.

Each course utilized these channels slightly differently. In Course B, student teams used their Microsoft Teams channels exclusively for collaboration on the team project; the instructor had access to the channels, but posted to them only to answer questions when explicitly tagged by a team member. In contrast, student teams in Course A used their Slack channels both for team collaboration, and to respond to discussion prompts in some course assignments.

Student teams in both courses were expected to employ sound agile software engineering practices emphasized in each course, including the use of issues, Kanban (project) boards, feature branches, pull requests, code reviews, automated tests, and retrospectives. However, grading of students and teams differed significantly between the two courses. In Course A, teams were awarded a grade based on the number of story points they completed during the term, with teaching personnel determining the story point values of each team's completed issues. In contrast, teams in Course B were graded based on a structured rubric, which the instructor used to perform a detailed evaluation of teams' GitHub repositories, chat channels, and deployed software. Individual grades were computed from team grades by applying individual multipliers derived from peer evaluation surveys [32] administered after each sprint.

3.3 Data Collection and Analysis

Teams' chat messages were collected through Slack (Course A) and Microsoft Teams (Course B). At the end of each course, we exported all teams' chat channel messages to a spreadsheet for analysis—one message per row. The original corpus included 6,811 messages (posts and replies). To keep the focus of the analysis on team communications surrounding their software development activities, we eliminated from the original corpus messages posted by non-student participants (teaching staff and bots). In addition, in Course A, class assignments occasionally gave student teams participation credit to engage in chat channel discussions that were not directly related to their software projects. We also eliminated the messages prompted by those assignments. After eliminating these messages, we arrived at a corpus of 4,915 messages.

We used the coding scheme of Curtis and Lawson [15] as a starting point for our content analysis. Through preliminary efforts to categorize our chat messages using the framework's five categories (Planning, Contributing, Seeking Input, Reflection, and Social Interaction), we changed the categories as follows:

- Planning was renamed to Planning/Coordination to capture episodes in which team members either plan or coordinate joint activities.
- Seeking Input was renamed to Seeking Technical Info *&* Assistance to capture team members' efforts not only to obtain technical (e.g., coding or tool) information, but also to solicit assistance in addressing technical questions or problems.
- Emotional Response was added to capture team members' expressions of gratitude, regret, appreciation, pride, and acknowledgment of others' work—expressions that were not included in the original framework of Curtis and Lawson [15].

Table 3 presents our modified content coding scheme. Messages were coded into multiple categories when applicable, and in the order in which they were originally posted, so that context could be considered. Through multiple iterations of coding small samples of messages and discussing the results, we gradually refined a coding manual that clarifies the coding categories through detailed examples and guidelines. Using that manual, two analysts independently

Table 2: Key Dimensions of Team Projects

Project Dimension	Course A	Course B
Project Focus	Existing legacy code projects	Self-selected with option to choose from list of recommended projects
Team Formation	Based on CATME survey, teams randomly assigned to projects	Self-selected; unassigned students randomly assigned to teams
Team size	5-6	3-5
Starting code base	Code base from existing project	Code base from full stack app developed in first part of course
Project Length (Weeks)	3-4	5
Collaboration Tools Used	GitHub, Slack	GitHub, Microsoft Teams
Number of sprints	3	4

Table 3: Definitions of Content Categories

Category/Code		Definition	Examples
Planning/ Coordination	P	Asking a question or making a statement/proposal about how the team should organize or coordinate its work: What's in scope/out of scope, who is on the team, who does what, in what order, and when, etc.	"Can you meet tomorrow at 10 am?" "Let's break this user story into two." "Should we break this user story into two?" "Jon, can you do the pull request for that issue?"
Contributing	C	Providing information, stating opinions, asking questions, answering questions, or performing work.	"I have created a PR for that issue." "I think you need to refactor that code."
Seeking Technical Info & Assistance	S	Seeking technical information and assistance regarding programming, software architecture, development tools (e.g., Visual Studio Code, GitHub), communication tools (e.g., Slack), and software development processes (e.g., how retrospectives work).	"Are there examples of peer reviews?" "How do I get access to the database?" "Can you help me with Slack's attachment feature?"
Reflection	R	Reflecting on or assessing individual or team processes and strategies. May suggest what to stop doing, start doing, or continue doing, but does not include a definite timeline for changes.	"I could have done that differently." "All of us working on this isn't a good use of time." "I think it's really worth reevaluating our priorities."
Social Interaction	I	Asking about or sharing personal information unrelated to the group task for the purpose of building social relationships.	Sharing of a meme or joke Personal introductions
Emotional Response	E	Expressing gratitude or acknowledging the work of others.	"Great job fixing that bug!"

coded a 14% sample of our corpus (n = 692). The analysts achieved 89% overall agreement, with a Cohen's Kappa value of 0.84. Having established high inter-rater reliability, the two analysts each coded half of the remaining messages.

To better understand teams' use of reflection, we performed a follow-up analysis of chat vignettes that included messages coded as Reflection. For this analysis, we considered all Reflection messages (n=63) and the five messages that preceded and followed them. This yielded a corpus of 358 messages.

In the first pass of the follow-up analysis, we used the surrounding context of each Reflection message to derive the five categories described in Table 4. These categories identify what we observed to have prompted the Reflection messages. In a second pass, we coded each Reflection message into one of those categories. Since the corpus of Reflection messages was relatively small (n=63), we opted not to verify inter-rater reliability. Instead, we had three of the authors code all messages independently, compare results, and resolve disputes through further discussion.

4 RESULTS

4.1 Message Content

On average, each team posted 214 messages to their chat channels (SD = 162) during the project. Figure 1 presents a stacked bar chart comparing message content by team. Bars are shaded to indicate the percentage of messages in each category. Although Figure 1 seems to show a similar categorical pattern across all teams, a chi-squared test of homogeneity indicates that teams differed significantly with respect to the categorical distribution of their chat messages (df = 110, $\chi^2 = 554.3, p < 0.001$).

Table 4: Definitions of Prompt Categories

Category/Code		Definition
Course Requirement	R	Prompted by a course requirement such as the need to perform a retrospective
Team Activity	A	Prompted by the team's naturally occuring software development activities
Presenting Demo	D	Prompted by presenting a required software demo
Receiving Grade	G	Prompted by receiving a team project grade

Visual inspection of Figure 1 shows that across all teams, online chats were dominated by two content categories: Contribution (M = 43%, SD = 11%) and Planning/Coordination (M = 43%, SD = 11%). This suggests that teams tended to appropriate their team chat channels to plan and coordinate their collaboration on the project, and to update each other on their progress.

Of the remaining 14% of teams' chat messages, 10% were divided between Emotional response (M = 5%, SD = 2%) and Social Interaction (M = 5%, SD = 4%). These types of messages served to build team rapport and camaraderie by sharing personal information, injecting humor into the process, and expressing empathy and understanding for each others' situations.

The remaining four percent of the chat messages were equally divided between Seeking Technical Info & Assistance (M = 2%,

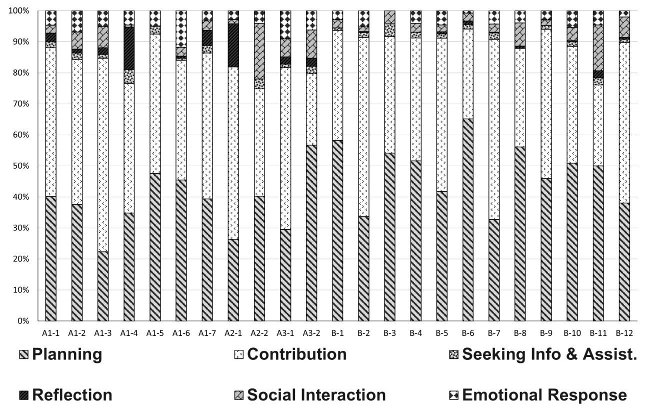

Figure 1: Message Content by Team

SD = 1%) and REFLECTION (*M* = 2%, *SD* = 4%). These types of messages were relatively rare.

4.2 A Closer Look at Reflection

Across the entire corpus, 63 messages were coded as REFLECTION. Table 5 presents the number of REFLECTION messages posted by each team, broken down by the four prompt categories described in Table 4. The bottom row (**% of all chat**) of Table 5 shows the percentage of each team's chat messages coded as REFLECTION.

As Table 5 shows, messages prompted by COURSE REQUIREMENTS dominated teams' REFLECTION messages, composing 76% (48/63) of the REFLECTION corpus. Further analysis of messages coded as COURSE REQUIREMENTS indicates that nearly all (46 of 48, 96%) were prompted by the requirement to perform a *retrospective*—an agile practice in which teams identify what went well, what needs improvement, and what can be changed going forward [24]. In Course A, eight of 11 teams engaged in a retrospective through their chat channel; two of 12 teams did so in Course B.

Another 10% of REFLECTION messages (6 of 63) were prompted by course events outside of teams' software development process. The RECEIVING GRADE prompt led two teams to assess why they got the grade they did. In Course B, teams were required to present a demo of their software to the instructor at the end of each sprint. In response to the PRESENTING DEMO prompt, one team assessed how their demo went on two separate occasions.

The remaining 14% of REFLECTION messages (9 of 63) arose organically within teams' software development process (TEAM ACTIVITY). For instance, upon completing their third sprint, a member of Team B5 observed that "we really need that core functionality that makes that app what it is." They went on to reflect on the team's process: "I think maybe it's worth reevaluating our priorities." This led to a planning conversation about how to better allocate the team's time in the next sprint.

In another example of reflection prompted by TEAM ACTIVITY, the leader of Team B10 became frustrated by their perception that other team members were not contributing: "You didn't deliver anything in terms of documentation, code, or really communication when *name redacted* asked for progress updates this morning. This is completely unacceptable." The team leader used this reflection on team process to propose a detailed plan for how to complete the remaining parts of the sprint.

5 DISCUSSION

While teams varied widely both in the extent to which they used their chat channel and in the categorical frequency of their message content, two content categories dominated all teams' chats: PLANNING/COORDINATION and CONTRIBUTION. With respect to our research question, we found that teams rarely used their chat channels to engage in reflection: *just two percent of teams' chat messages were coded as REFLECTION.* Moreover, REFLECTION messages that arose organically during the course of teams' development processes were even rarer: just 14% of the REFLECTION messages were prompted by TEAM ACTIVITY; the rest were prompted by course requirements or events.

One explanation of this finding is that students did not feel comfortable sharing their reflections in their team chat channels. To engage in reflection, students must assess themselves and their team. This requires interpersonal risk, which in turn requires psychological safety [21] that may not have been present in the team chat channels. Our findings suggest that students were rarely willing to take the risk. When they did, it was most often prompted by the requirement to engage in retrospectives, which led to 46 out of the 63 REFLECTION messages (73%) observed in the chat corpus.

Given this, rather than requiring teams to perform retrospectives only periodically, instructors could encourage increased reflection by requiring teams to perform so-called *instant retros* on a daily basis—a practice advocated by the agile community [30]. In an instant retro, any time a team member notices something about an individual or team process that is going well or not going well, the team member posts the observation to the team chat channel for further discussion and possible action.

In addition to influencing student behavior through course requirements, instructors can play a powerful role in establishing the psychological safety necessary for students to engage in more frequent reflection [21]. They can do this by, for example, modeling reflection, being vulnerable themselves, and providing students with compelling reasons to engage in reflection [21].

Instructors might also encourage reflection through explicit learning activities. For instance, instructors could have students read about the role and importance of reflection in software development, and then lead a class activity where teams are presented with hypothetical team development issues to reflect on and reason about. Teams could share their reflections and the ways those reflections might lead to changes, thus seeing for themselves the value of reflection in the software development process.

Teams' interactions with communication (e.g., Slack, Microsoft Teams) and collaboration (e.g., GitHub) tools provide further opportunities to prompt for reflection. For instance, a bot could monitor a team chat channel for PLANNING/COORDINATION and CONTRIBUTION messages that provide opportunities for reflection, and generate appropriate prompts. The same could be done in response to GitHub events such as commits, issue creations, pull requests, and the outcomes of automated test runs.

Table 5: Reflection Message Counts by Team, Broken Down by Prompt Category

Prompt Category	A1-1	A1-2	A1-3	A1-4	A1-5	A1-6	A1-7	A2-1	A2-2	A3-1	A3-2	B-1	B-2	B-3	B-4	B-5	B-6	B-7	B-8	B-9	B-10	B-11	B-12
Course Requirement	4	4	5	6	4	5	6	0	0	0	6	0	2	0	0	3	0	0	2	0	0	1	0
Team Activity	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0	1	0	0	2	0	3
Presenting Demo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0
Receiving Grade	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0
Total Count	**4**	**4**	**5**	**6**	**4**	**5**	**6**	**0**	**0**	**1**	**6**	**0**	**2**	**0**	**0**	**9**	**2**	**1**	**2**	**0**	**2**	**1**	**3**
% of all chat	2%	3%	14%	3%	1%	2%	14%	0%	0%	1%	5%	0%	0%	0%	0%	1%	1%	0%	1%	0%	0%	2%	1%

6 THREATS TO VALIDITY

6.1 Internal Validity

Internal validity concerns the extent to which our research methods uncover the truth regarding our research question in our specific population. One such threat is that we have looked for evidence of reflection only in the messages that students post in the team channels of the official course chat app. We know that our students used other means to communicate, including Discord, Zoom, and direct messages (DMs) within our official chat apps; however, we did not have access to those communications. Other places where students may exhibit reflection, but that our analysis missed, include commit messages and comments on GitHub issues and pull requests; analyzing these could provide a fertile area for future work.

Another threat to internal validity is that during one course instance in the study, the instructor invited teams that were finished with a particular assignment to do a "mini retrospective" and post their reflections in the chat. Teams that were not finished were given extra time to complete that assignment. This resulted in some teams having the opportunity for "prompted reflection", while other teams did not. We have mitigated this threat by categorizing those messages into the Requirements category so that they can be distinguished from reflections that arose organically.

6.2 External Validity

External validity considers the extent to which our results can be generalized to software engineering education or other team software development contexts. A threat to external validity is that all of the course offerings in the study occurred during academic year 2020–2021, the first full academic year of the global Covid-19 pandemic. During this period, instruction was fully online in the courses studied. The additional stress of this change, along with the external stressors of the pandemic itself, may have influenced the outcomes in ways that will not be apparent without further follow-up study under non-pandemic conditions.

7 SUMMARY AND FUTURE WORK

To explore undergraduate software teams' use of reflection, we have presented the results of a content analysis of their chat channels. The results show that evidence of reflection in chat messages is rare. When it does occur, it is most often in response to course requirements, rather than arising organically through ongoing software development work. Based on these findings, we have identified ways that computing instructors can prompt for reflection in team projects through pedagogical and technological interventions.

To study and improve teams' use of reflection in software projects, we would like to explore three avenues of future work. First, to better understand teams' use of reflection during the software development process, we would like to triangulate data from multiple sources, including chat channels, video meeting recordings, and team and individual interviews. Considering a broader range of places where reflection can occur in the software development process will not only increase the validity of future studies, but also provide greater insight into how teams use reflection and what further opportunities might exist to improve it.

Second, we would like to implement a series of interventions targeted at improving reflection in the team software development process. These include (a) integrating explicit instruction on reflection into our courses, and (b) integrating reflection software prompts into team collaboration and communication tools. In quasi-experiments, we would like to gauge the extent to which these and other interventions are effective in improving reflection.

Third, while reflection is known to be valuable to the software development process, we would like to be able to tie it to specific team outcomes. For instance, does higher use of reflection correlate with better team grades, higher quality software processes and products, or better learning attitudes? In addition, longitudinal studies could provide insight into whether students' use of reflection in undergraduate team projects is associated with students being better prepared for jobs in the software profession.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under grant nos. 1915196 and 1915198.

REFERENCES

[1] Ken Abernethy and Kevin Treu. 2009. Teaching Computing Soft Skills: An Experiential Approach. *J. Comput. Sci. Coll.* 25, 2 (dec 2009), 178–186.

[2] R. R. Adisurya, H. B. Santoso, S. Fadhilah, and O. Lawanto. 2020. Information visualization of metacognitive skills during the software development process based on an adapted engineering design metacognitive questionnaire. *Journal of Physics: Conference Series* 1566, 1 (June 2020), 012078. https://doi.org/10.1088/1742-6596/1566/1/012078 Publisher: IOP Publishing.

[3] Rana Alkadhi, Teodora Lata, Emitza Guzmany, and Bernd Bruegge. 2017. Rationale in Development Chat Messages: An Exploratory Study. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE/ACM, Buenos Aires, Argentina, 436–446. https://doi.org/10.1109/MSR.2017.43

[4] Agile Alliance. 2022. What is a Retrospective? https://www.agilealliance.org/glossary/heartbeatretro/

[5] Andrew Begel and Beth Simon. 2008. Novice software developers, all over again. In *Proceedings of the Fourth international Workshop on Computing Education Research (ICER '08)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/1404520.1404522

[6] Andrew Begel and Beth Simon. 2008. Struggles of new college graduates in their first software development job. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education (SIGCSE '08)*. ACM, New York, NY, USA, 226–230. https://doi.org/10.1145/1352135.1352218

[7] Elizabeth Bjarnason, Anne Hess, Richard Berntsson Svensson, Björn Regnell, and Joerg Doerr. 2014. Reflecting on Evidence-Based Timelines. *IEEE Software* 31, 4 (2014), 37–43. https://doi.org/10.1109/MS.2014.26

[8] Lil Blume, Ron Baecker, Christopher Collins, and Aran Donohue. 2009. A "Communication Skills for Computer Scientists" Course. *SIGCSE Bull.* 41, 3 (jul 2009), 65–69. https://doi.org/10.1145/1595496.1562903

[9] Quincy Brown, Frank Lee, and Suzanne Alejandre. 2009. Emphasizing Soft Skills and Team Development in an Educational Digital Game Design Course. In *Proceedings of the 4th International Conference on Foundations of Digital Games* (Orlando, Florida) *(FDG '09)*. ACM, New York, NY, USA, 240–247. https://doi.org/10.1145/1536513.1536557

[10] Kevin Buffardi. 2020. Assessing Individual Contributions to Software Engineering Projects with Git Logs and User Stories. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, Portland, OR, 650–656.

[11] Christopher N. Bull and Jon Whittle. 2014. Supporting Reflective Practice in Software Engineering Education through a Studio-Based Approach. *IEEE Software* 31, 4 (2014), 44–50. https://doi.org/10.1109/MS.2014.52

[12] Lori Carter. 2011. Ideas for Adding Soft Skills Education to Service Learning and Capstone Courses for Computer Science Students. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. ACM, New York, NY, USA, 517–522. https://doi.org/10.1145/1953163.1953312

[13] Emily Laue Christensen and Maria Paasivaara. 2022. Learning Soft Skills through Distributed Software Development. In *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering* (Pittsburgh, PA, USA) *(ICSSP'22)*. ACM, New York, NY, USA, 93–103. https://doi.org/10.1145/3529320.3529331

[14] Michelle Craig, Phill Conrad, Dylan Lynch, Natasha Lee, and Laura Anthony. 2018. Listening to early career software developers. *Journal of Computing Sciences in Colleges* 33, 4 (April 2018), 138–149.

[15] David Curtis and Michael Lawson. 2019. Exploring Collaborative Online Learning. *Online Learning* 5, 1 (2019), 21–24. https://doi.org/10.24059/olj.v5i1.1885

[16] Roger T. Johnson David W. Johnson. 2007. *Cooperation and the Use of Technology.* Routledge, New York, NY, Chapter 33, 785––811. https://doi.org/10.4324/9780203880869.ch33

[17] Joanna F. DeFranco and Philip A. Laplante. 2017. Review and Analysis of Software Development Team Communication Research. *IEEE Transactions on Professional Communication* 60, 2 (2017), 165–182. https://doi.org/10.1109/TPC.2017.2656626

[18] Vladan Devedzic, Bojan Tomic, Jelena Jovanovic, Matthew Kelly, Nikola Milikic, Sonja Dimitrijevic, Dragan Djuric, and Zoran Sevarac. 2018. Metrics for Students' Soft Skills. *Applied Measurement in Education* 31, 4 (2018), 283–296. https://doi.org/10.1080/08957347.2018.1495212 arXiv:https://doi.org/10.1080/08957347.2018.1495212

[19] Tania Mara Dors, Frederick M. C. Van Amstel, Fabio Binder, Sheila Reinehr, and Andreia Malucelli. 2020. Reflective Practice in Software Development Studios: Findings from an Ethnographic Study. In *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, Munich, Germany, 1–10. https://doi.org/10.1109/CSEET49119.2020.9206217

[20] Tore Dybå, Neil Maiden, and Robert Glass. 2014. The Reflective Software Engineer: Reflective Practice. *IEEE Software* 31, 4 (2014), 32–36. https://doi.org/10.1109/MS.2014.97

[21] Amy C. Edmondson. 2003. *Managing the Risk of Learning: Psychological Safety in Work Teams.* John Wiley & Sons, Ltd, West Sussex, England, Chapter 13, 255–275. https://doi.org/10.1002/9780470696712.ch13 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470696712.ch13

[22] Anatolij Fandrich, Nils Pancratz, and Ira Diethelm. 2022. Soft Skills and Technical Competence: Interdisciplinary Qualification of First-Year Computer Science Students. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 2* (Dublin, Ireland) *(ITiCSE '22)*. ACM, New York, NY, USA, 637. https://doi.org/10.1145/3502717.3532146

[23] Jack W. Fellers. 1996. Teaching Teamwork: Exploring the Use of Cooperative Learning Teams in Information Systems Education. *SIGMIS Database* 27, 2 (apr 1996), 44–60. https://doi.org/10.1145/243350.243359

[24] Francino, Yvette and Denman, James. 2021. What is an Agile retrospective? https://www.techtarget.com/searchsoftwarequality/definition/Agile-retrospective

[25] Wouter Groeneveld, Brett A. Becker, and Joost Vennekens. 2020. Soft Skills: What Do Computing Program Syllabi Reveal About Non-Technical Expectations of Undergraduate Students?. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. ACM, New York, NY, USA, 287–293. https://doi.org/10.1145/3341525.3387396

[26] Orit Hazzan. 2002. The reflective practitioner perspective in software engineering education. *Journal of Systems and Software* 63, 3 (2002), 161–171. https://doi.org/10.1016/S0164-1212(02)00012-2

[27] Orit Hazzan and Gadi Har-Shai. 2013. Teaching Computer Science Soft Skills as Soft Concepts. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. ACM, New York, NY, USA, 59–64. https://doi.org/10.1145/2445196.2445219

[28] Christopher Hundhausen, Adam Carter, Phillip Conrad, Ahsun Tariq, and Olusola Adesope. 2021. Evaluating Commit, Issue and Product Quality in Team Software Development Projects. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*. ACM, New York, NY, USA, 108–114. https://doi.org/10.1145/3408877.3432362

[29] Indeed Editorial Team. 2021. 11 Important Soft Skills for Software Developers. https://www.indeed.com/career-advice/career-development/software-developer-soft-skills

[30] Dominic Krimmer. 2019. The Instant Retrospective. https://luis-goncalves.com/instant-retrospective/ Section: Agile Retrospectives Ideas: Games For Your Next Retrospective.

[31] K. Krippendorff. 1980. *Content analysis: an introduction to its methodology.* Sage Publications, Beverly Hills.

[32] Richard Layton, Matthew Ohland, and Hal R Pomeranz. 2007. Software for Student Team Formation and Peer Evaluation: CATME Incorporates Team-Maker. In *2007 Annual Conference & Exposition*. ASEE, Honolulu, Hawaii, 12–1286. https://peer.asee.org/software-for-student-team-formation-and-peer-evaluation-catme-incorporates-team-maker

[33] Andrew C. Loignon, David J. Woehr, Jane S. Thomas, Misty L. Loughry, Matthew W. Ohland, and Daniel M. Ferguson. 2017. Facilitating Peer Evaluation in Team Contexts: The Impact of Frame-of-Reference Rater Training. *Academy of Management Learning & Education* 16, 4 (Dec. 2017), 562–578. https://doi.org/10.5465/amle.2016.0163 Publisher: Academy of Management.

[34] Dastyni Loksa, Lauren Margulieux, Brett A. Becker, Michelle Craig, Paul Denny, Raymond Pettit, and James Prather. 2021. Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use. *ACM Trans. Comput. Educ.* TBD, TBD (dec 2021), 1–30. https://doi.org/10.1145/3487050 Just Accepted.

[35] Anna B. Marques, Bruna Ferreira, Adriana Lopes, and Williamson Silva. 2020. Stimulating the Development of Soft Skills in Software Engineering Education through Design Thinking. In *Proceedings of the 34th Brazilian Symposium on Software Engineering* (Natal, Brazil) *(SBES '20)*. ACM, New York, NY, USA, 690–699. https://doi.org/10.1145/3422392.3422488

[36] Janet Metcalfe and Arthur P. Shimamura (Eds.). 1994. *Metacognition: Knowing about knowing.* The MIT Press, Cambridge, MA, US. https://doi.org/10.7551/mitpress/4561.001.0001 Pages: xiii, 334.

[37] Tom Nurkkala and Stefan Brandle. 2011. Software Studio: Teaching Professional Software Engineering. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. ACM, New York, NY, USA, 153–158. https://doi.org/10.1145/1953163.1953209

[38] Daniela Pedrosa, Mario Madureira Fontes, Tânia Araújo, Ceres Morais, Teresa Bettencourt, Pedro Duarte Pestana, Leonel Morgado, and José Cravino. 2021. Metacognitive challenges to support self-reflection of students in online Software Engineering Education. In *2021 4th International Conference of the Portuguese Society for Engineering Education (CISPEE)*. CISPEE, Lisbon, Portugal, 1–10. https://doi.org/10.1109/CISPEE47794.2021.9507230

[39] James Prather, Brett A. Becker, Michelle Craig, Paul Denny, Dastyni Loksa, and Lauren Margulieux. 2020. What Do We Think We Think We Are Doing? Metacognition and Self-Regulation in Programming. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20)*. ACM, New York, NY, USA, 2–13. https://doi.org/10.1145/3372782.3406263

[40] Sadhana Puntambekar, Gijsbert Erkens, and Cindy Hmelo-Silver. 2011. *Analyzing Interactions in CSCL: Methods, Approaches and Issues.* Springer, New York, NY. https://doi.org/10.1007/978-1-4419-7710-6

[41] Sandeep Purao and Hoi Suen. 2010. Designing a Multi-Faceted Metric to Evaluate Soft Skills. In *Proceedings of the 2010 Special Interest Group on Management Information System's 48th Annual Conference on Computer Personnel Research on Computer Personnel Research* (Vancouver, BC, Canada) *(SIGMIS-CPR '10)*. ACM, New York, NY, USA, 88–91. https://doi.org/10.1145/1796900.1796934

[42] A. Radermacher and G. Walia. 2013. Gaps between industry expectations and the abilities of graduates. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 525–530. https://doi.org/10.1145/2445196.2445351

[43] Harry Budi Santoso, Oenardi Lawanto, Betty Purwandari, R. Yugo K. Isal, and Rian Fitriansyah. 2017. Investigating Students' Metacognitive Skills while Working on Information Systems Development Projects. In *2017 7th World Engineering Education Forum (WEEF)*. IEEE, Kuala Lumpur, Malaysia, 478–483. https://doi.org/10.1109/WEEF.2017.8467121

[44] D. Schön. 1983. *The reflective practitioner: How professionals think in action.* Basic Books, New York.

[45] J.E. Sims-Knight and R.L. Upchurch. 1998. The acquisition of expertise in software engineering education. In *FIE '98. 28th Annual Frontiers in Education Conference. Moving from 'Teacher-Centered' to 'Learner-Centered' Education. Conference Proceedings (Cat. No.98CH36214)*, Vol. 3. IEEE, Tempe, AZ, USA, 1302–1307 vol.3. https://doi.org/10.1109/FIE.1998.738679

[46] Kathleen Swigger, Matthew Hoyt, Fatma Cemile Serçe, Victor Lopez, and Ferda Nur Alpaslan. 2012. The temporal communication behaviors of global software development student teams. *Computers in Human Behavior* 28, 2 (2012), 384–392. https://doi.org/10.1016/j.chb.2011.10.008