Thematic analysis of reflective peer feedback in programming-heavy engineering courses

Abhimanyu Ghosh
Engineering & Technology Department
University of Wisconsin-Stout
Menomonie, Wisconsin, USA
ghosha@uwstout.edu

Abhishek Verma
Engineering & Technology Department
University of Wisconsin-Stout
Menomonie, Wisconsin, USA
vermaa@uwstout.edu

***Abstract*—The work presented in this Full Paper is categorized as Innovative Practice, as per FIE guidelines. This project is based on data collected from two coding-intensive courses in an engineering program at a mid-sized, open-enrollment primarily undergraduate institution (PUI). Notably, the engineering curriculum does not include an introductory programming course.**

In both courses, groups of students worked on engineering problems using code, after which they presented their initial drafts to the entire class. Their peers, watching the presentations, were asked to provide written feedback in the form of 4 objective questions and 3 short, essay-type responses.

A thematic analysis of the essay-type responses indicates that the prompts determined what the responses were likely to focus on. When asked what the presenters had done well, the responses overwhelmingly related to the quality of the presentation itself without details on content. When asked for what could be improved, the responses most often included checkbox-like statements like whether the answer was correct, if there were any syntax errors, etc.

When asked for one takeaway from the presentation they just watched in a live classroom, responses in both classes revolved around coding-related terms and problem-solving steps, relevant to each course. When classes went online for Fall 2020 and presentations were held over video conferencing, the responses to this prompt also included significantly more expressions of emotions like commiserating with peers or being validated by them. Themes emerging from the other two prompts did not vary based on class modality.

This thematic analysis can, therefore, inform instructors teaching programming-heavy engineering courses on how best to design reflective activities that are integrated into course assessments.

***Index Terms*—reflection, oral presentation, baccalaureate institutions, peer review**

I. Introduction

A project on code presentation and accompanying surveys was originally started with the intention of improving student learning outcomes in programming-heavy science and engineering classes at the authors' institution, a teaching-focused, mostly undergraduate university. Over three semesters in a pilot study, the method of presentation, group formation, and comment prompts in the survey were improved based on data gained from each semester. At the end of the pilot, a semester-long project was run in Fall 2019 in two separate courses. This was repeated in Fall 2020 in the same courses, and then in completely online format due to Covid-19 restrictions. Results from the two courses, in each of the two semesters, are analyzed in this paper.

Most qualitative research focuses on introductory programming classes meant for college freshmen [1], [2], or computer science (CS) majors in later classes [3]. The present work analyzed programming-heavy courses in the upper-level mechanical engineering (ME) curriculum. Moreover, standalone reflective activities tend to trigger negative reactions from students [4]. The authors, therefore, took care to design an assignment-integrated reflective activity to possibly mitigate any negative reactions.

In both courses, groups of students presented their work on biweekly assignments, that asked them to solve engineering problems using code, to the class. Their peers were asked to fill out a 5-minute survey, listed in the appendix, with both objective (Yes/No) questions and prompts for short, essay-type comments. The objective questions were meant to be pointers only, since their accuracy cannot be gauged in the absence of the instructor's own judgement of the initial draft. Responses to the three essay-type comments formed the student reflection data on which the authors performed thematic analysis.

The paper is divided into eight different sections, this being the first. The second section presents a review of the institutional context and existing literature while the third one details the participants, process, and the survey instrument. Next, the fourth section describes the method by which thematic analysis was done, the results of which are discussed in the fifth and sixth sections corresponding to the two research questions presented below. Finally, sections seven and eight discuss the implications of the present work and directions of future work.

A. Research Questions

This work intends to investigate two primary research questions from the student reflection data collected.

1) What are the major themes arising out of student reflections, written as feedback for code presentation by their peers, in programming-heavy engineering courses?
2) How do the themes differ by prompt, course, and class modality?

II. Context

In this section, we describe the significance of the study for programming courses in engineering curricula, specifically in open-enrollment, less-selective institutions. We also talk about the general trends found in literature of the pedagogy of programming-intensive coursework, the need for a more qualitative approach when studying programming courses in engineering, and review the existing scholarship on peer review and thematic analysis in programming courses.

A. Institution and Curriculum

In order to fully understand the context of this project and its significance, it is important to discuss the type of institution where the project was carried out. The authors' university is a mid-sized, open-enrollment, 4-year institution in upper Midwestern United States. Most programs in the university have a focus on applied learning and practical training. The university itself is a primarily undergraduate institution (PUI), with faculty expected to devote at least 70% of their time toward teaching.

While individual programs at the institution, like ME, may have entry requirements like minimum ACT scores, prior experience with high-school level physics, etc., the student body itself is not the outcome of a highly selective process. This is noteworthy since a highly selective admissions process often leads to a student body with prior experience that is not necessarily captured by 4-year program plans, like high school calculus courses, AP classes, etc. For example, Bound et al. [5] show data from the National Center for Education Statistics that as far back as 2004, 52.3% of students attending a selective private institution had taken high school calculus as opposed to only 23.3% of all students attending 4-year institutions.

Therefore, the conclusions presented can have important implications for other PUIs, community colleges, 2-year technical colleges, and other similar institutions.

B. Focus on the Quantitative

Programming-intensive courses have a large proportion of student work in easily quantifiable formats. As a result, a lot of research on the pedagogy of such courses focuses heavily on quantifiable student work.

Macfadyen et al. [6] analyzed data from a Learning Management System (LMS) to identify 15 variables that demonstrated correlations with the students' final grades in the class. Watson, Li, and Godwin [7] made use of the WatWin system to log student response times to various errors, improving on the method of calculating an Error Quotient pioneered by Jadud [8]. Becker [9] introduced a new parameter, Repeated Error Density, based on the number of consecutively repeated errors and proved it to be more context-independent. Carter et al. [10] proposed a more holistic system of using a normalized Programming State Model to observe not just syntactic errors made during compilation but also semantic runtime errors.

However, in all the above cases and many others using similar tools, the focus is primarily on

- introductory programming courses, and
- syntax and compilation errors, not so much logical ones.

For courses in non-CS STEM programs that include a high amount of programming, instructors often focus on the implementation of mathematical logic in code and errors related to the same. Eliminating syntax and structural errors, while still crucial, is not the sole learning outcome of such coursework. As such, simply logging errors and analyzing them is unlikely to paint the entire picture of student learning in such courses.

C. Presentation as a Form of Peer Review

Code review by peers is widely used in professional code development but their use in coding-intensive courses is still limited, although growing [11]. As scientists already familiar with the professional peer review process in research, many instructors implement similar processes in the classroom with a view to improving student learning [12]. But most recent studies on peer reviews in the classroom for programming-related courses have fallen short of rectifying concerns that are often specific to such courses such as the ease of plagiarism [13], inflated scores in reviews that did not reflect student performance in exams [14], reluctance to criticize fellow students [15] in the interests of maintaining a non-judgemental atmosphere [16] etc.

Trytten [17] followed a peer review process in their programming-based course, evolving over multiple semesters, that mitigates a lot of the usual concerns but relies heavily on instructor feedback, barely qualifying as peer review. Further, the objective nature of the responses expected in the reviews limits its application, as discussed in Section II-B

The pilot study preceding this project [18] sought to develop classroom presentation to peers, followed by a feedback survey, as a credible alternative to peer reviews without the usual concerns. The results of that study informed the design of the current project.

D. Thematic Analysis of Student Reflections

When applied on reflective statements from students, thematic analysis is a powerful tool *"for identifying, analyzing and reporting patterns (themes) within data"* [19]. Pedrosa et al. [20] performed thematic analysis of weekly forms submitted by students regarding their strategies for self-regulated learning, in a second-year undergraduate course in an Informatics Engineering program. Boyer and Celepkolu [1] used thematic analysis to determine student sentiments toward pair programming and the cognitive, affective and social factors emerging out of students' reflection essays in an introductory programming course. Cain and Woodward [2] *"applied thematic analysis to the reflective reports presented by students as part of their portfolio submission for an introductory programming unit"*.

In all the above-mentioned studies,

- the courses selected were either in introductory programming or in majors closely related to computer science, and
- student reflections were collected through standalone activities.

There is evidence, however, that standalone reflective activities tend to trigger negative reactions from students, even when the positive effects are acknowledged. Martins et al. [4] write that *"... most students had a negative initial reaction when told that they would have to write a reflection every two weeks. However, in most cases they later reached the conclusion it was positive."*.

George [21] asked students to make weekly journal entries in their programming course. When asked to rate the experience, *"The quantitative response was rather negative with very few students rating the journal well...Many students appeared positive in the comments they wrote but would give the journal a 'not useful' rating in quantitative terms."*

III. Setup

The current project was set up in Fall 2019 keeping in mind, therefore, the need for qualitative analysis in engineering courses that are not specifically *introduction to programming*, of reflective pieces that are preferably not generated from standalone activities. The setup is described here in detail, with regards to the participants, the code presentation process that worked as an alternative to peer review, and the survey instrument used to extract reflective data for analysis. The project was implemented in Fall 2019 and Fall 2020 semesters, in the same two courses.

A. Participants

The two courses that participated in the current project are:

- Course A: Numerical Methods in Engineering
- Course B: System Dynamics

1) Course A: Numerical Methods in Engineering: This is often the first programming-related course that a lot of mechanical engineering (ME) undergraduates take at the authors' university. There is no introductory programming course in the curriculum prior to this, so students need to be introduced to basic programming concepts in Python at first. This creates a fast-paced, intense course that takes students from basic programming all the way through solving first and second order ordinary differential equations near the end, to fulfill course objectives.

2) Course B: System Dynamics: This is a course that students take in their final year of study, often during their last semester. They know the basics of programming and have used code-based problem-solving in 2-3 courses prior to this. The coding process in this course is less traditional and more related to arranging system circuits, tracing flow of information, etc. using MATLAB Simulink toolbox.

TABLE I
Number of Students

	Course A		Course B	
	Participants	**Modality**	**Participants**	**Modality**
Fall 2019	20	Face-to-face	22	Face-to-face
Fall 2020	25	Online	19	Online

B. Code Presentation Process

In each course, students were divided up into groups of 3-4 to work on homework assignments. On average, there were 24-30 students in each of these courses every semester and only half the groups were required to present during any given session.

Each group of students had 10 minutes to explain how their code was solving a given problem, after which the non-presenting students filled out a 5-minute survey, as discussed in Section I. This process was repeated for all the presenting groups, at the end of which the responses were compiled and sent to respective groups as feedback.

This presentation-and-feedback activity was designed, as mentioned in Section II-C, as an alternative to peer review that reduces problems like plagiarism, reluctance to criticize, etc. The authors realize, however, that this cannot replace a thorough peer review in every situation given the time constraints of an in-class activity. Treating the feedback as reflective pieces for further analysis was also aimed at preventing negative reactions to reflective pieces, as discussed in Section II-D.

While all students were asked to fill out the surveys to help their peers with feedback, only data from those who had consented is used for the purposes of this work. The Institutional Review Board at the university had verified the project before waiving any need for oversight. The number of consenting students and course modalities are listed in Table I.

C. Survey

The survey questionnaire was same throughout the project, with slight modifications in the words depending on the course. The feedback survey filled out by students after every presentation is added at the end of this paper as an appendix. Of special importance are the three comment prompts listed below for quick reference, since the responses to these comment prompts form our data set for analysis.

- **Prompt 1**: Comment on the things that you think the presenters had done well, in writing the code they presented.
- **Prompt 2**: Comment on the things that the presenters can do to improve their work.
- **Prompt 3**: What is a takeaway from this presentation that you believe will be useful to you while working on your own homework attempt?

TABLE II
NUMBER OF RESPONSES

	Fall 2019		Fall 2020	
	Course A	**Course B**	**Course A**	**Course B**
Prompt 1	247	177	715	143
Prompt 2	243	154	682	134
Prompt 3	237	154	669	128

The number of responses to each prompt, in each course, is listed in Table II to help gauge the size of our data set.

D. Group formation

Group formation among students was informed by observations summarized in [18]. In both courses, both semesters, students were allowed to form their own groups based on best practices reported in literature. Harding [22] proposed a solution called *flocking* in which students are guided into forming groups based on schedules and commitments, allowing them to self-select while reducing free-riding. Mello [23] detailed a method to improve accountability in self-selected groups by making students ask each other about expected grades, schedules, commitment to the course, etc.

Carefully-planned group formation methods ensured student buy-in into a project that relied heavily on publicly presenting their assignment as a group. No major problems or complaints were encountered regarding group formation in any of the courses.

IV. Method

Thematic analysis is a qualitative analytic method that can be used to identify patterns in written information. This is widely used in analyzing student reflections, as Section II-D demonstrates.

Braun and Clarke [19] describe a six-phase accessible approach to apply thematic analysis to any set of data, in and beyond psychology. The authors have followed the same in this project, resulting in the themes listed and described in Table III.

A. Familiarizing yourself with your data

After every set of presentations, survey responses were organized by presenting groups and sent back to them as feedback. At the end of each semester, responses were downloaded and categorized by each student taking the survey and consenting to their data being used in research. During both these processes, the authors gained initial familiarity with the data.

B. Generating initial codes

Survey data, categorized by consenting students and prompts at the end of the semester, was gathered in a spreadsheet. The authors then ran together through every comment, treated as a data extract, assigning one or more initial codes to each. This process was done by the authors jointly, meeting together, and an expansive approach was adopted. When they came up with different codes and could not agree on dropping one, both codes were assigned to the comment. This biases the overall data set toward a higher count for every theme.

This was, by far, the most time-consuming exercise in the whole process, with the highest probability of subjective interpretation. However, with both authors assigning codes jointly, there was less chance of personal prejudice creeping into the assignment of codes.

C. Searching for themes

Once the initial codes were generated for all the comments in the entire project (two courses in each of two semesters), each code was assigned one or more sub themes. The sub themes were assigned on the basis of the content of the initial codes, rather than the sentiment behind them. The list of sub themes were separated by semester, with only slight differences between them in terms of uniqueness.

D. Reviewing themes

The sub themes collected from both semesters were categorized into larger themes, depending on their context and meaning. This process required concept maps to visualize and required a few short iterations, before the authors agreed on the final configuration. Random data extracts (comment samples) were selected to verify if a sub theme belonged to the appropriate theme. A given sub theme, it was decided, could only belong to a single theme. This counteracted, to a certain extent, the bias toward a higher count caused by the expansive approach mentioned in Section IV-B.

Significantly, the authors stuck to an initial decision to reach agreement on each code and theme *"through collaborative discussion rather than independent corroboration"*, as recommended by Smagorinsky [24], from step B through D.

E. Defining and naming themes

The themes that emerged out of the last step were named, based on what they were meant to describe and the contexts in which they were found in the data set. These names and descriptions can be found in Table III.

F. Producing the report

The work presented in this paper consists of the last phase of thematic analysis, as described by Braun and Clarke [19].

V. Results: Themes

This section describes the themes that arise out of the analysis of comment responses, in an attempt to answer the first research question from Section I-A. The following lines provide explanations of the kinds of responses that were categorized under each theme, their representative samples, and also the notable characteristics of each.

A. Themes: Descriptions

The entire dataset of responses could be categorized into eight themes, following the method outlined in Section IV. The themes are numbered and described in Table III for quick reference.

1) Judging the presentation: This theme dominates all responses to the first prompt, in both courses and modalities, whether counting the number of responses or the number of students who responded. When asked about the positive aspects of code being presented, it seems that the overwhelming response is to talk about the presentation itself, the ease of understanding, verbal and written explanations of code, etc. An example of a student comment falling under this theme, from prompt 1 in course A (Fall 2019) is as follows:

> *"The presenters wrote the code so that it is very easy to read and everything is very organized in functions"*

The frequency of appearance of this theme goes down substantially from prompts 1 through 3, when counting the number of responses but less so when counting the number of students. This may indicate that there are a few students who simply respond to the presentation itself, as opposed to the code, no matter what they are asked. For example, a response to prompt 3, asking for a takeaway, in course A (Fall 2019) generates this:

> *"how much comments help when looking at others code"*

2) Making a checklist: This is a theme that makes substantial appearances in every prompt, whether counting responses or students. There is no specific trend in its appearances, whether looking at different prompts, courses or modalities, except its high frequencies.

In the context of a short presentation of complex-looking code, this follows expectations that responding students will often look for error messages, answers, specific values, etc., anything that is similar to checking boxes. The yes/no questions in the survey, prior to the comment prompts, may also have played a role in priming them to respond in a similar fashion. A response to prompt 1 in course B (Fall 2019) ran:

> *"They had it the code running. The for loop ran without any errors"*

A similar response to prompt 3 in course A (Fall 2020) ran:

> *"From their answer mine seems correct"*

3) Problem-solving process: This is the theme that appears most often in responses to prompt 3, whether in terms of number of responses or students. When asked for a takeaway from the presentation, the student responding thinks much more frequently about debugging their code, verifying their solutions, the preparatory work needed prior to writing code, etc., generating more technical phrases, problem-solving concepts, etc. Here is an example of this from prompt 3 in course A (Fall 2020):

> *"A good takeaway is that they had a different Taylor's series equation and I can take that into account"*

Most students seem to have responses correlating to this theme in all prompts; it is only that those types of responses increase in frequency in prompt 3. Here is a relevant example from prompt 2, asked to point out what could be improved in the presenters' code, in course A (Fall 2020):

> *"Make a for loop from 200 to 800 and multiply that by the resistance matrix to make the code run all of the resistance values"*

The phrases themselves change with the course but the theme stays the same. Below is an example from prompt 1, when asked what the presenters did well, in course B (Fall 2019):

> *"I like that pieces of Governing EQs are all grouped near each other and well spaced from other groups. Good placement of stars, gives good frame of reference for how gravity is affecting the mass."*

4) Presenters' demeanor: Not too many responses fall under this theme. However, students whose responses do are much more likely to be in course A than in course B. Responses under this theme focus on the presenting group's overall demeanor, their perceived teamwork, their willingness to answer questions and receive feedback, etc.

The difference between the two courses may suggest that the relative inexperience of students in course A, in presenting to an audience, causes them to both applaud and criticise their peers more frequently. However, the frequencies of such responses are too low to decipher any trends.

Here is a student responding positively to prompt 1 in course A (Fall 2019):

> *"Even though they knew their data was not correct, they still presented what they had and took help from the class well. They showed that they wanted the assistance and were willing to admit that their code was not correct."*

Here is another response from the same student in the same course, to prompt 2:

> *"I think that all members of the group should present a little bit of work. Even if they present some small amount of info, it shows participation."*

5) Learning resources: This is a theme that encompasses responses referring to tutors, feedback or talking points from the instructor, examples demonstrated in class, or time management. There are very few responses in this category in Course B. In Course A, while quite a few students mention this theme, the number of responses is still fairly low. An example of this from prompt 3 in course A (Fall 2020) would be:

> *"the conversation with [the instructor] was helpful because of the problems I am also having"*

6) Expressing emotional state: A certain portion of responses primarily relate to commiserating with the presenters or being inspired by them; being thankful for the presentations or be confused by them. There are two major trends in this theme. Firstly, the proportion of such responses goes up substantially in online modality, for both courses taught by different instructors (the authors of this paper). This is a student comment to prompt 3 in the online version of course A (Fall 2020):

> *"I think him and I are actually at the same standstill with this problem so I'm not sure I gained any*

TABLE III
THEMES AND DESCRIPTIONS

Theme number	Theme	Description
1	Judging the presentation	Responses relate to how the work is laid out, whether it is easy to follow, if the presenters did a good job explaining their work.
2	Making a checklist	Responses provide a checklist of whether certain conditions are fulfilled. E.g. is the code correct, is it running without errors, does it have errors, is it complete. Responses also focus on output from the code, whether those are values, plots, errors in output, or anything related to units.
3	Problem solving process	Responses talk about details of code, mention debugging and troubleshooting, talk about presenters' knowledge of coding, how to verify and what to change in the code. Responses may also emphasize the details in the problem statement, equations in the assignment, work necessary to prepare for writing code, and other preliminaries.
4	Presenters' demeanor	Responses focus on the presenting group's participation and teamwork, their willingness to answer questions, receive feedback and admit mistakes, their overall attitude and confidence, adherence to ethical principles.
5	Learning resources	Responses refer to the tutor, the instructor, and course material.
6	Expressing emotional state	Responses focus on the emotional state of the person providing feedback. This can be commiserating with the presenters, expressing gratitude, being inspired, encouraged or confused, expressing difficulty in completing the assignment, discussing own plans and prior work.
7	Unsubstantial	Responses have no substantial content, or are not related to the prompt, or just mention "nothing".
8	Sense of community	Responses talk about the utility of class discussions during presentations, assistance received by presenters from their classmates, and general usefulness of looking at different presentations.

knowledge from this. But it was reassuring to know that someone is in the same spot as me."

Secondly, this trend is noticeable in the number of responses but not necessarily in the number of students, indicating that the same students may be expressing their emotional states a lot more in the online format, when prompted for a takeaway. An very typical example of this is shown here, taken from responses to prompt 3 in course B (Fall 2020), again in the online format:

"We all kind of struggled"

7) Unsubstantial: A significant proportion of responses, for all prompts, were unsubstantial and grouped under this theme. They either contained phrases cheering their peers like "good job", "well done", etc. or mentioned "nothing", "looks good", etc., especially when asked to figure out what could be improved, in prompt 2. Sometimes, they did not seem related to the prompt at all. There are no clear trends in this theme, even as some sections have more such responses than others.

8) Sense of Community: A tiny proportion of responses talked about the utility of the code presentations, assistance received from peers, and the general usefulness of looking at a variety of work on the same topic. There are no clear trends in this theme, however. Here are two responses to prompt 3, from different students, in course A (Fall 2020):

"I think it's helpful to see other people's attempts at problems as it helps me fix my own"

An example from prompt 3 in course A (Fall 2020) is below:

"Helpful to see an example of the code done by others"

B. Themes: Frequencies

As can be seen in Section V-A, not all themes find equal representation in the student reflection data. As indicators of their relative popularity, the frequencies of each theme, separated by prompt and course, are shown as a percentage of total responses on the left part of Fig. 1. The authors remind the reader here that a single comment can fall under multiple themes, due to being assigned multiple initial codes as outlined in Section IV-B.

In addition, higher frequency of a certain theme is no guarantee of its spread among a significant portion of the surveyed students. Many students have comments that overwhelmingly fall under a certain theme while others are more balanced. Here are five different responses from a student to prompt 1 that demonstrates this.

"Very good code flow"
"Very organized code flow"
"The code is easy to follow the flow"
"The code has good flow and is easy to follow"
"The code was easy to read and follow"

To get a more representative picture of the prevalence of a theme among students in the class, frequencies of themes are shown as a percentage of students on the right hand side of Fig. 1. E.g. if 50% of students in a given course have responses falling under a certain theme (even just a single response), the vertical bar for that course goes up to 50%.

VI. RESULTS: THEME VARIATIONS

Attempting to answer the second research question (Section I-A), four key findings come into focus when the themes

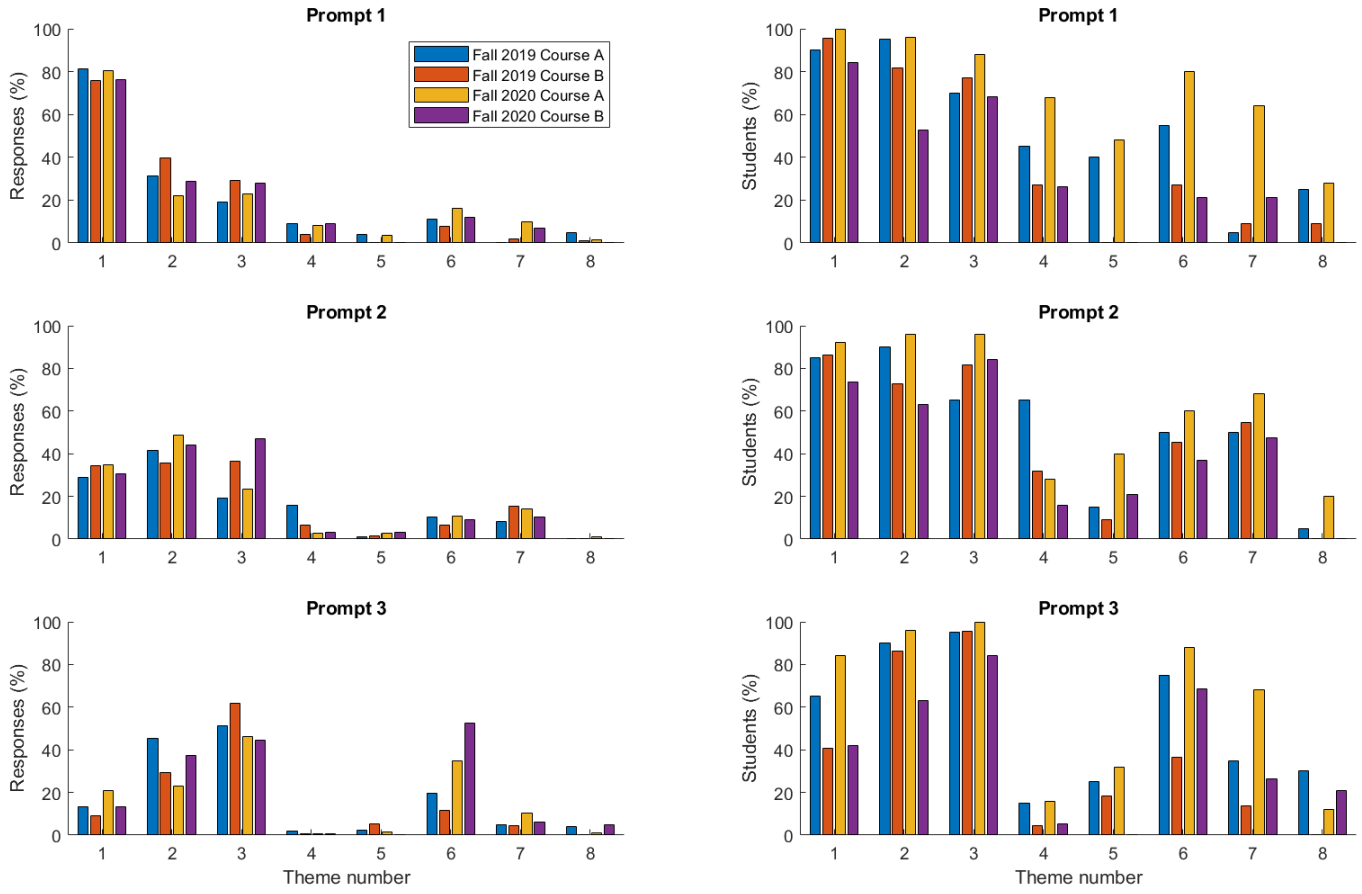


Fig. 1. Frequency tables of responses and students under themes 1 through 8, for each prompt and course.

are analyzed for patterns in how and where they differ.

1) **Themes 1 and 2 are the most prevalent overall**
When asked for feedback on code presentations, most responses in any class are concentrated on themes 1 and 2, talking about the presentation and explanation, and checking boxes for answers, values, syntax errors, etc., respectively. This may indicate that these engineering students' inexperience with coding causes them to focus on aspects of the presentation they are more comfortable judging. Peer reviews and presentations in similar courses, therefore, should take this into account when designing feedback instruments.
2) **Theme 3 comes into focus in prompt 3**
Theme 3, dealing with technical details of coding like debugging, logical and syntactical errors with details, preparatory work prior to coding, etc., usually shows highest frequencies of any theme in prompt 3, both in terms of responses and students. This verifies findings of the pilot study [18] and suggests, in combination with the first key finding, that students in non-CS programming courses may only feel comfortable talking about coding and code-based problem-solving when it is related to them (as opposed to their peers) and new learning experiences (as opposed to judging correctness of existing assignments).
3) **Theme 1 towers over others in prompt 1**
Applause for presenting students happens to be primarily about presentation, explanation and ease of reading, even though reviewing students can clearly make out the finer points of code and problem-solving, as evidenced by prevalence of theme 3 in responses to prompt 3.
These students, thus, may not feel competent enough to know that certain code and problem-solving methods are indeed correct, leading them to talk more about aspects they feel more comfortable judging, whether or not this is actually true. This may be a corollary to research in CS courses that found scores received in peer reviews to be far higher than actual competencies [14], because the peer review scores are based more on presentation than content.

4) **Students express their emotional states more in the online format**
Responses in both courses for the Fall 2020 semester, when they were conducted online, show a spike in frequencies of Theme 6 in prompt 3 i.e. far more responses were expressing students' emotional states when asked for a takeaway. Comments talking about how difficult a problem was, how they felt hopeful looking at another group's presentation, how everyone in the class struggled with a particular problem, etc. were more common. There was no observable change in the number of students responding this way; it is that those students who did respond this way were doing so more frequently.
The authors suspect that reduced social presence and sense of community in online learning environments [25], more so during the Covid-19 pandemic when so many traditional students were forced to take online classes, may have contributed to a stronger need for validation and commiseration among some students. Interestingly, the lack of responses falling under Theme 8 (Sense of community) shows that many students may not have been conscious of this need to connect with others.

VII. Conclusions

The authors' primary objectives, as outlined in the research questions, were to qualitatively analyze peer feedback from code presentations for prominent themes and to figure out how the themes varied with feedback prompts, courses and modalities. As outlined in Table III, student responses were categorized into eight different themes. In Section V-A, each theme is discussed in detail, with representative responses.

In this section, the authors discuss the implications of the themes that emerged and the variations in where and when they appear.

A. Implications for engineering students in programming courses

The key findings in how the appearance of the themes varies (Section VI) illustrate that themes related to judging the presentation, checklist responses, and the problem-solving process were the most prominent, though differing in how frequently they appeared in a given prompt.

Engineering students seem unsure of their ability to judge the technical aspects of coding in others' work. They tend to focus on readability, presentation and explanation when asked about the work of their peers while talking far more about problem-solving and coding when asked what they themselves learned from that work. However, these students felt comfortable judging their peers' work where they could quantify it or compare it against a checklist (such as answers, code free of syntax errors, plots that could be verified, etc.).

B. Implications for traditional students in online courses

One of the key findings that sprung a surprise on the authors is the increase in responses that primarily expressed the responding student's emotional state, when the courses were held completely online during Fall 2020. As discussed in Section VI, these responses often commiserated with the presenters or expressed relief at being validated by the presenters' work. The authors hypothesize that this may be due to reduced social presence and a sense of community in these classes, due to a number of possible factors. While the authors have no data to substantiate this, it may be worthwhile in the future to look into the role played by peer reviews and presentations to increase social presence in online classes.

VIII. Future Directions

The results of this thematic analysis attempt to provide an overall picture of what engineering students think of when they look at their peers presenting code. As the next step, individual students' responses could be analyzed thematically to investigate any correlations with attainment of learning outcomes. This would require:

- New machine learning algorithms in Natural Language Processing (NLP) to thematically categorize student responses, making the process faster and less labor-intensive. Existing commercial software like *https://getthematic.com/insights/thematic-analysis-software/* can also be used, as long as the algorithm training procedures are open and transparent.
- Implementing the project in multiple classrooms, preferably in multiple sections of the same course, with clearly-defined experimental and control groups. Any improvement in learning outcomes from the presentation-and-feedback process is likely to be smaller in scale compared to the effects of course content, instructor capability, type of institution, etc., necessitating tightly-controlled distinctions between the experimental and control groups.

In addition, student attitudes toward integrating reflective activities into peer feedback may also be analyzed vis-a-vis standalone reflective pieces, to further investigate possibilities of reducing negative reactions as those outlined in Section II-D.

Finally, repeating similar studies on student populations in STEM disciplines in PUIs, community colleges, 2-year technical colleges will help gauge the breadth of validity of these findings.

Acknowledgment

The authors thank all the students who consented to have their feedback data be used for research, as well as the Nakatani Teaching and Learning Center (NTLC) at UW-Stout who helped initiate this project.

References

[1] M. Celepkolu and K. E. Boyer, "Thematic analysis of students' reflections on pair programming in CS1," *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, vol. 2018-Janua, pp. 771–776, 2018.

[2] A. Cain and C. J. Woodward, "Examining student reflections from a constructively aligned introductory programming unit," *Conferences in Research and Practice in Information Technology Series*, vol. 136, pp. 127–136, 2013.

[3] W. Yan-qing, L. Yi-jun, M. Collins, and L. Pei-jie, "Process improvement of peer code review and behavior analysis of its participants," *SIGCSE'08 - Proceedings of the 39th ACM Technical Symposium on Computer Science Education*, pp. 107–111, 2008.

[4] S. W. Martins, A. J. Mendes, and A. D. Figueiredo, "Student reflections as an influence in the dynamics of an introductory programming course," *Proceedings - Frontiers in Education Conference, FIE*, pp. 1–6, 2011.

[5] J. Bound, B. Hershbein, and B. T. Long, "Playing the admissions game: Student reactions to increasing college competition," *Journal of Economic Perspectives*, vol. 23, no. 4, pp. 119–146, 2009.

[6] L. P. Macfadyen and S. Dawson, "Mining LMS data to develop an "early warning system" for educators: A proof of concept," *Computers and Education*, vol. 54, no. 2, pp. 588–599, 2010.

[7] C. Watson, F. W. Li, and J. L. Godwin, "Predicting performance in an introductory programming course by logging and analyzing student programming behavior," *Proceedings - 2013 IEEE 13th International Conference on Advanced Learning Technologies, ICALT 2013*, pp. 319–323, 2013.

[8] M. C. Jadud, "Methods and tools for exploring novice compilation behaviour," *ICER 2006 - Proceedings of the 2nd International Computing Education Research Workshop*, vol. 2006, no. Figure 1, pp. 73–84, 2006.

[9] B. A. Becker, "A new metric to quantify repeated compiler errors for novice programmers," *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, vol. 11-13-July, pp. 296–301, 2016.

[10] A. S. Carter, C. D. Hundhausen, and O. Adesope, "The normalized programming state model: Predicting student performance in computing courses based on programming behavior," *ICER 2015 - Proceedings of the 2015 ACM Conference on International Computing Education Research*, pp. 141–150, 2015.

[11] C. Bauer, K. Figl, M. Derntl, P. P. Beran, and S. Kabicher, "The student view on online peer reviews," *ACM SIGCSE Bulletin*, vol. 41, no. 3, p. 26, 2009.

[12] N. M. Trautmann, "Designing Peer Review for Pedagogical Success: What Can We Learn from Profe...: BartonPlus," *Journal of College Science Teaching*, vol. January, 2009.

[13] Y. Wang, H. Li, Y. Feng, Y. Jiang, and Y. Liu, "Assessment of programming language learning based on peer code review model: Implementation and experience report," *Computers and Education*, vol. 59, no. 2, pp. 412–422, 2012.

[14] X. Li, "Using Peer Review to Assess Coding Standards - A Case Study," *Proceedings. Frontiers in Education. 36th Annual Conference*, pp. 9–14, 2006.

[15] T. McMahon, "Peer feedback in an undergraduate programme: Using action research to overcome students' reluctance to criticise," *Educational Action Research*, vol. 18, no. 2, pp. 273–287, 2010.

[16] T. Papinczak, L. Young, and M. Groves, "Peer assessment in problem-based learning: A qualitative study," *Advances in Health Sciences Education*, vol. 12, no. 2, pp. 169–186, 2007.

[17] D. A. Trytten, "A design for team peer code review," *ACM SIGCSE Bulletin*, vol. 37, no. 1, p. 455, 2005.

[18] A. Ghosh and D. Sinkovits, "Classroom presentation of code: An alternative peer review process," in *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling '19. New York, NY, USA: Association for Computing Machinery, 2019.

[19] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[20] D. Pedrosa, J. Cravino, L. Morgado, and C. Barreira, "Self-regulated Learning in Computer Programming: Strategies Students Adopted During an Assignment," in *Communications in Computer and Information Science*, 2016, vol. 2, pp. 87–101.

[21] S. E. George, "Learning and the Reflective Journal in Computer Science," *Journal Australian Computer Science Communications*, vol. 24, no. 1, pp. 77–86, 2002.

[22] L. M. Harding, "Students of a Feather " Flocked " Together : A Group Assignment Method for Reducing Free-Riding and Improving Group and Individual Learning Outcomes," *Journal of Marketing Education*, vol. 40, no. 2, pp. 117–127, 2018.

[23] J. A. Mello, "Improving Individual Member Accountability in Small Work Group Settings," *Journal of Management Education*, vol. 17, no. 2, pp. 253–259, 1993.

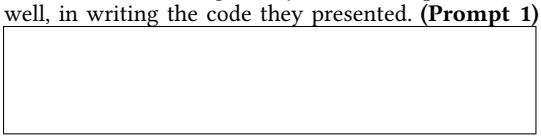
[24] P. Smagorinsky, "The method section as conceptual epicenter in constructing social science research reports," *Written Communication*, vol. 25, no. 3, pp. 389–411, 2008.

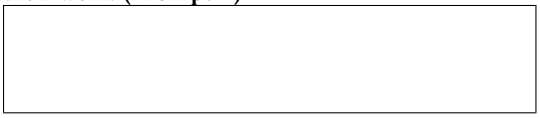
[25] Y. J. Park and C. J. Bonk, "Synchronous learning experiences: Distance and residential learners' perspectives in a blended graduate course," *Journal of Interactive Online Learning*, vol. 6, no. 3, pp. 245–264, 2007.

Appendix

Following is the full survey that students in courses A and B were asked to fill out at the end of each presentation. The first four questions are only meant to be pointers and reflective data was collected from questions 5 - 7.

1) Does the code answer all parts of the question asked in the homework (relevant to coding)?
   - ☐ YES
   - ☐ NO
2) Does the code run without syntax errors when executed (even if the answers are incorrect)?
   - ☐ YES
   - ☐ NO
3) Is the code written in a way that makes it easy for you to understand it without asking questions (formatting, white space, comments etc.)?
   - ☐ YES
   - ☐ NO
4) Does the code follow suggested 'good practices' like making functions, storing numbers in arrays, writing comments explaining their work, initializing variables when used the first time etc.?
   - ☐ Mostly follows the 'good practices' taught in class
   - ☐ Follows 'good practices' sometimes, but not always or even mostly
   - ☐ Barely follows any 'good practices'
5) Comment on the things that you think the presenters had done well, in writing the code they presented. **(Prompt 1)**

   
6) Comment on the things that the presenters can do to improve their work. **(Prompt 2)**

   
7) What is a takeaway from this presentation that you believe will be useful to you while working on your own homework attempt? **(Prompt 3)**