SE 611 : Software Metrics

Project Name: GQM (Goal Question Metrics)

Submitted by

Muktadul Islam: BSSE 1215

Rufidatul Radium: BSSE 1226

Ahnaf Mubashshir Mobin: BSSE 1232

Supervised By

Emon Kumar Dey

Associate Professor

Institute of Information Technology, University of Dhaka

Submission Date: 18/09/2023

Table of Contents

1. GQM

GQM (Goal-Question-Metric) is a framework used in software engineering and project management to set clear goals, formulate specific questions, and select relevant metrics for evaluating and improving software projects and processes. It promotes data-driven decision-making and continuous improvement by systematically assessing progress and outcomes.

1. **Goal Definition:** GQM begins with setting clear and measurable goals for a project or process. These goals serve as the foundation and overarching objectives.

2. **Question Formulation**: The framework then formulates specific questions related to each goal. These questions help in understanding whether the goals are being achieved and guide the measurement process.

3. **Metric Selection**: GQM identifies and selects appropriate metrics that can provide quantitative data to answer the formulated questions. Metrics serve as objective measures of progress and performance, facilitating data-driven decision-making.

2. Project Specification

2.1 Project Overview

The project seeks to improve code quality by evaluating and enhancing student programming practices in the context of SPL2, a curriculum at the institute that involves web technologies and software frameworks.

2.2 Motivation

The motivation behind this project is to elevate student programming practices in SPL2 by focusing on code quality. We are driven by the importance of code quality in the software industry, the need to provide real-world relevance to students, and the goal of fostering a data-driven, continuously improving educational environment. Ultimately, our motivation lies in ensuring student success, promoting educational excellence, and making a meaningful impact on the field of software engineering.

2.3 Scope

The project's scope is centered on a comprehensive evaluation and improvement of student programming practices in the SPL2 curriculum through the application of Smell Analysis. This entails gathering valuable data, conducting surveys, meticulously analyzing the collected information, and deriving curriculum enhancement recommendations. Furthermore, the project places a strong emphasis on fostering a culture of continuous improvement within the educational framework, ensuring that the benefits extend far beyond the immediate scope of the project.

3. Goal Specification

3.1 GQM Framework

General Statement: Enhancing Student Programming Practices in SPL2 through Smell Analysis.

3.2 PPE Approach

Purpose: To evaluate the student programming practice in SPL2 through Smell Analysis in order to enhance code quality.

Perspective: To evaluate the student programming practice in SPL2 through Smell Analysis in order to enhance code quality from the viewpoint of Software Engineers.

Environment: In IIT, SPL2 is part of the curriculum. Every student makes a project using web technologies or software frameworks. They are in a time constraint of six months. The goal is To evaluate the student programming practice in SPL2 through Smell Analysis in order to enhance code quality from the viewpoint of Software Engineers.

After specifying our goal through the purpose-perspective-environment framework, our final goal is-

“To systematically assess and improve student programming practices within the SPL2 curriculum by applying Smell Analysis techniques. This endeavor is driven by the aim to elevate the overall code quality, emphasizing a Software Engineer's perspective and accommodating the time-constrained environment inherent to the six-month duration of student projects at the Institute of Information Technology (IIT).”

3.3 Sub Goals

Our Goal has 6 more sub goals.

A. **Assessing Awareness of Code Smells:** Assess the students' familiarity with code smells before commencing their SPL2 projects, determining their pre-existing knowledge and awareness of this critical concept in software engineering.

B. **Identifying Code Smells in Student Projects:** Identify and categorize common code smells present in student projects developed during SPL2, shedding light on prevalent coding issues and their frequency of occurrence.

C. **Analyzing Code Quality Improvements:** Analyze whether students successfully address code smells in their projects, and ascertain the specific improvements in code quality resulting from these efforts, including readability, bug reduction, and performance enhancement.

D. **Evaluating the Impact of Code Smell Analysis:** Evaluate the overall impact of code smell analysis on students' programming practices, considering the perceived improvement from students' perspectives and the extent of formal training received.

E. **Gathering Feedback for Enhancement:** Collect valuable feedback and suggestions from students on how to improve the teaching of code smell analysis in SPL2, as well as the resources and tools they believe would enhance their learning experience.

F. **Assessing Time Management and Constraints:** Assess the effectiveness of students' time management during the six-month project development period, and identify specific time-related challenges they encounter, such as meeting deadlines and managing coursework.

3.4 Questions & Metrics

Sub-Goal A: Assessing Awareness of Code Smells

Q1: How familiar are students with the concept of code smells before starting their projects in SPL2?

M1. Pre-Project Familiarity Average Score

M2. Familiarity Distribution

Q2: Are students aware of common code smells that can affect code quality?

M1. Common Code Smell Awareness Ratio
M2. Level of Knowledge on Code Smells

Sub-Goal B: Identifying Code Smells in Student Projects

Q3: What are the most common code smells identified in the student projects

M1. Frequency of Common Code Smells

M2. Proportion of Projects with Each Common Code Smell

M3. Percentage of Projects with Long Methods

M4. Percentage of Projects with Large Classes

M5. Percentage of Projects with Duplicate Code

M6. Percentage of Projects with Magic Numbers

M7. Percentage of Projects with Feature Envy

M8. Percentage of Projects with Excessive Comments

Q4: How frequently do students encounter code smells in their projects?

M1. Code Smell Frequency

M2. Severity of Code Smells in Project

Sub-Goal C: Analyzing Code Quality Improvements

Q5: Have students successfully addressed code smells in their projects as part of their development process?

M1. Code Smell Addressing Success Rate

M2. Extent of Code Smell Addressing

Q6: What specific improvements in code quality have been observed as a result of addressing code smells?

M1. Observed Improved Readability

M2. Observed Fewer Bugs

M3. Observed Faster Execution

M4. Observed Easier Maintenance

Sub-Goal D: Evaluating the Impact of Code Smell Analysis

Q7: To what extent do students believe that analyzing code smells has improved their programming practices?

M1. Perceived Improvement Level

M2. Belief in Code Smell Analysis Impact

Q8: Have students received formal training or guidance on how to identify and address code smells in their projects?

M1. Formal Training Ratio

M2. Effectiveness of Training Perception

Sub-Goal E: Gathering Feedback for Enhancement

Q9: What suggestions do students have for improving the teaching of code smell analysis in SPL2?

M1. Most Suggested Enhancement Categories
M2. Frequency of Each Enhancement Suggestion

Q10: Are there specific resources or tools that students feel would help them better understand and address code smells?

M1. Most Desired Resources/Tools
M2. Frequency of Each Desired Resource/Tool

Sub-Goal F: Assessing Time Management and Constraints

Q11: How effectively do students manage their time during the six-month project development period in SPL2?

M1. Time Management Effectiveness
M2. Time Allocation Distribution

Q12: Are there specific time-related challenges that students face while working on their projects?

M1. Most Prevalent Time-related Challenges
M2. Frequency of Each Time-related Challenge

4. Questionnaire Preparation and Data Collection

In order to systematically evaluate and enhance student programming practices within the SPL2 curriculum through Smell Analysis, a comprehensive questionnaire has been designed. This questionnaire is a pivotal tool aimed at collecting valuable insights into students' familiarity with code smells, their awareness of common code smells, and their experiences in identifying and addressing code smells during their projects. The information gathered through this questionnaire aligns with the defined goals, questions, and metrics outlined in the GQM framework.

Key Breakdown of Surveys

To comprehensively capture the required data, the questionnaire is divided into sections, each addressing specific aspects related to code smells and their impact on the development process. The breakdown includes the following sections:

- Familiarity Assessment: Evaluating students' understanding of code smells and their familiarity with the concept.

- Awareness of Common Code Smells: Assessing students' knowledge of common code smells that can impact code quality.

- Encountering Code Smells: Exploring students' experiences regarding encountering code smells during project development.

- Approaches to Addressing Code Smells: Understanding the strategies employed by students to address and mitigate code smells in their projects.

Questionnaire Preparation and Data Collection Process:

- **Questionnaire Design**
  The design of the questionnaire is carefully tailored to address specific aspects related to code smells. It comprises structured and semi-structured questions, offering a versatile approach to gather quantitative and qualitative data. The questions are designed to encourage participants to reflect on their knowledge of code smells and their practical application in project development.

- **Questionnaire Sections**

  The questionnaire is organized into sections, each focusing on a particular facet of code smells and student practices:

  Taking a question for example:

  Q. Are students aware of common code smells that can affect code quality?

  1. Yes
  2. May Be
  3. No

  This question aims to assess the students' awareness of common code smells, which are indicators of potential issues affecting code quality. Understanding their awareness is vital for identifying areas where additional education or training may be needed. A 'yes' response indicates an existing level of awareness, while a 'no' or ‘may be’ response provides insight into the need for educational interventions.

- **Data Collection**

  Data for this questionnaire was collected through a secure and anonymous online survey platform. All participants are assured of confidentiality and privacy, encouraging open and honest responses. Additionally, a subset of respondents will be invited for semi-structured interviews to delve deeper into their experiences and perspectives on code smells.

Full questionnaires can be found here: https://forms.gle/EKnBCmfdSF9EFJov6

5. Data Visualization

Here we will represent the responses for each question in a percentage format to show the distribution of responses.

Q1

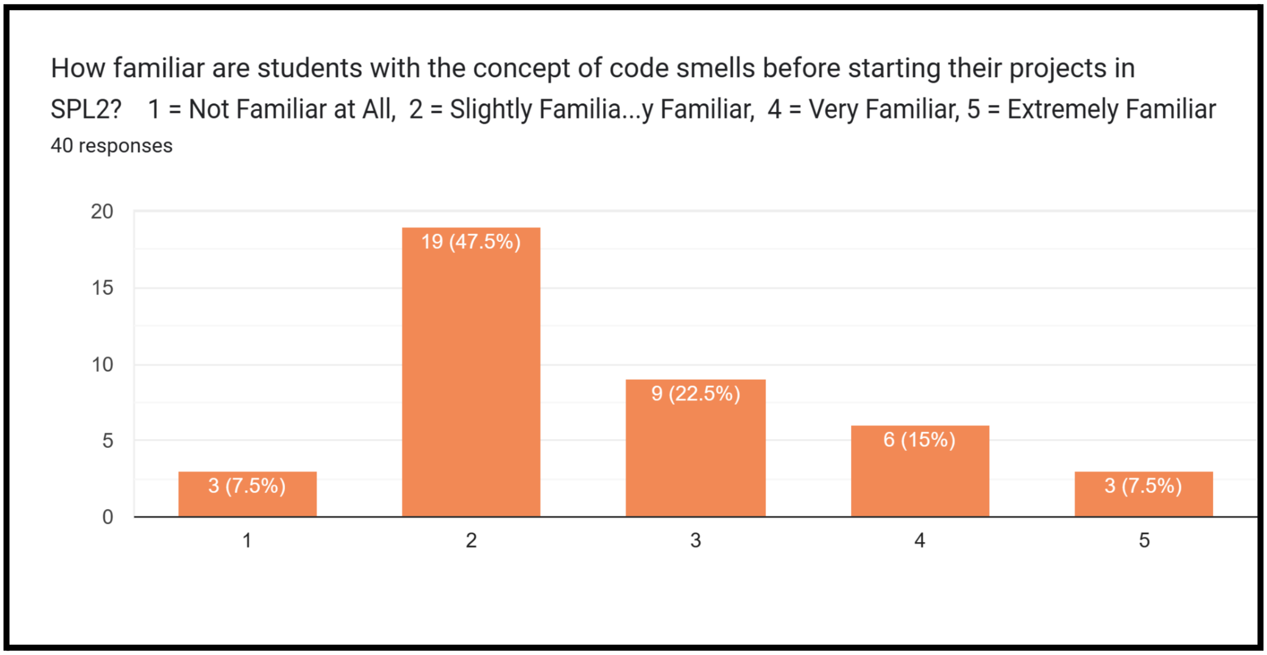

How familiar are students with the concept of code smells before starting their projects in SPL2? 1 = Not Familiar at All, 2 = Slightly Familia...y Familiar, 4 = Very Familiar, 5 = Extremely Familiar
40 responses

Figure 1: Respondents' Familiarity with Code Smells

Q2

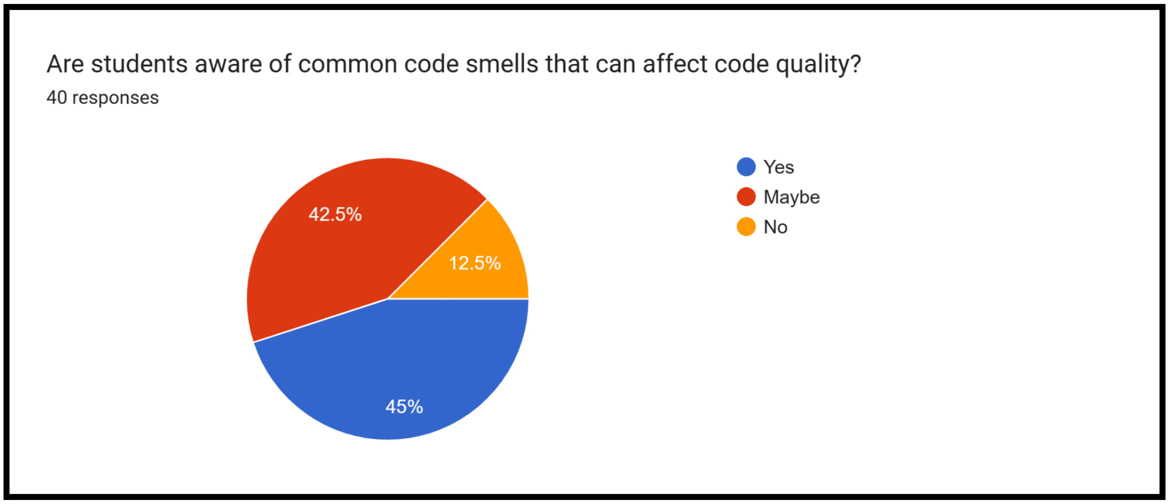
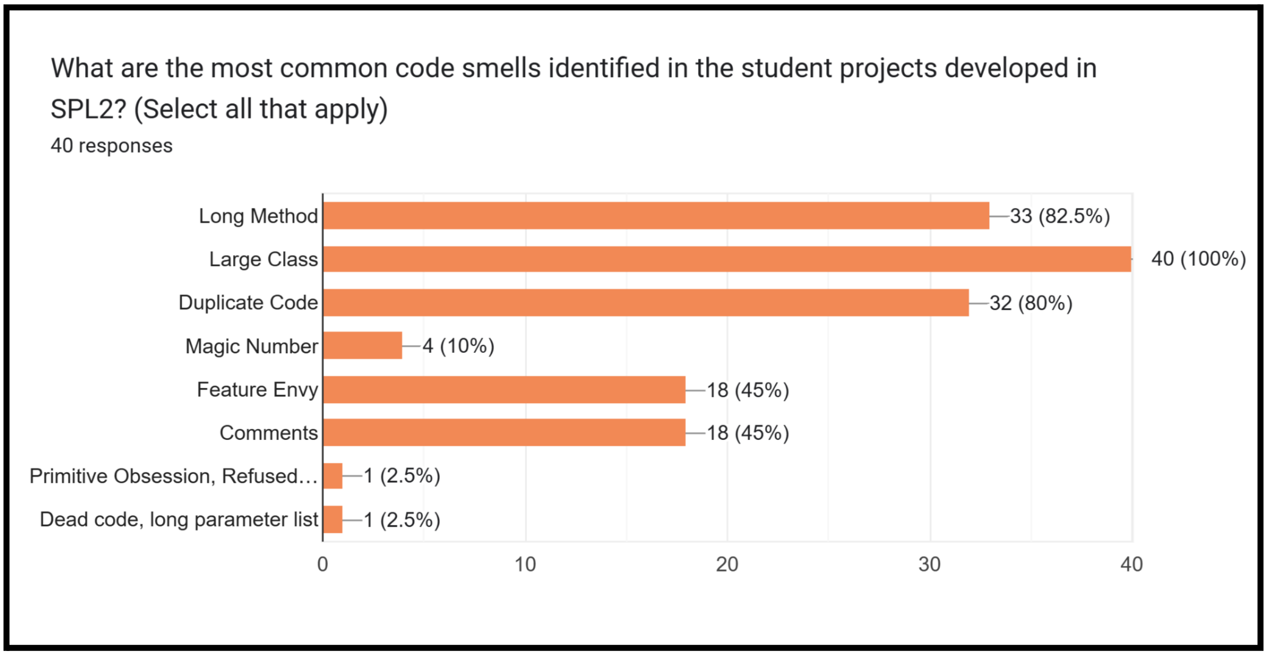

Are students aware of common code smells that can affect code quality?
40 responses

Figure 2: Respondents' Awareness of Common Code Smells

Q3

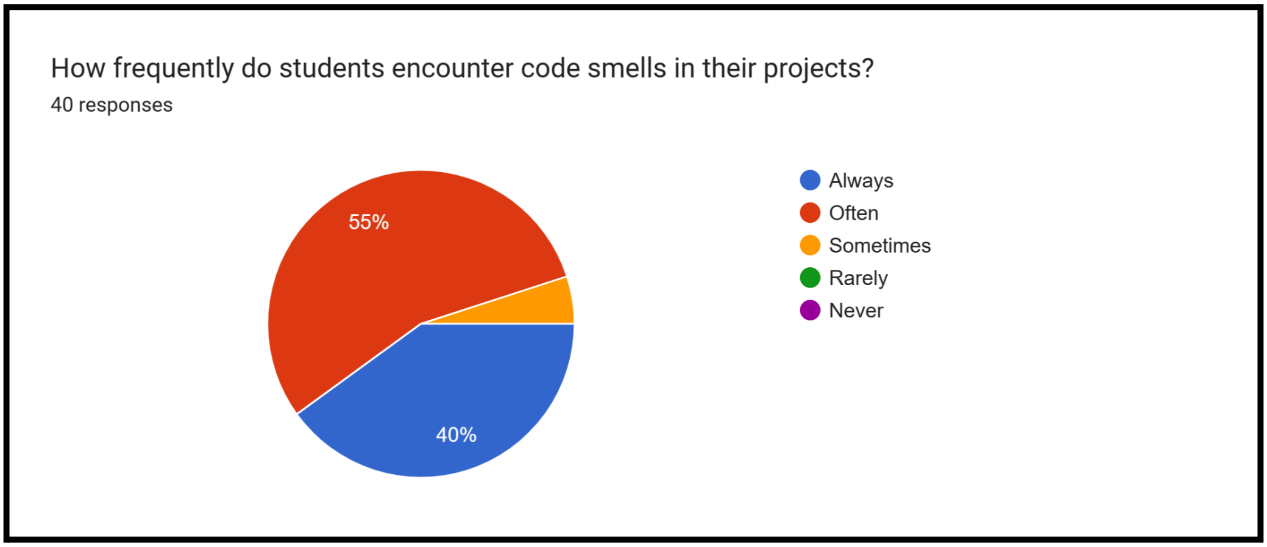


What are the most common code smells identified in the student projects developed in SPL2? (Select all that apply)

40 responses

Figure 3: Respondents' Identification of Code Smells in their Project

Q4

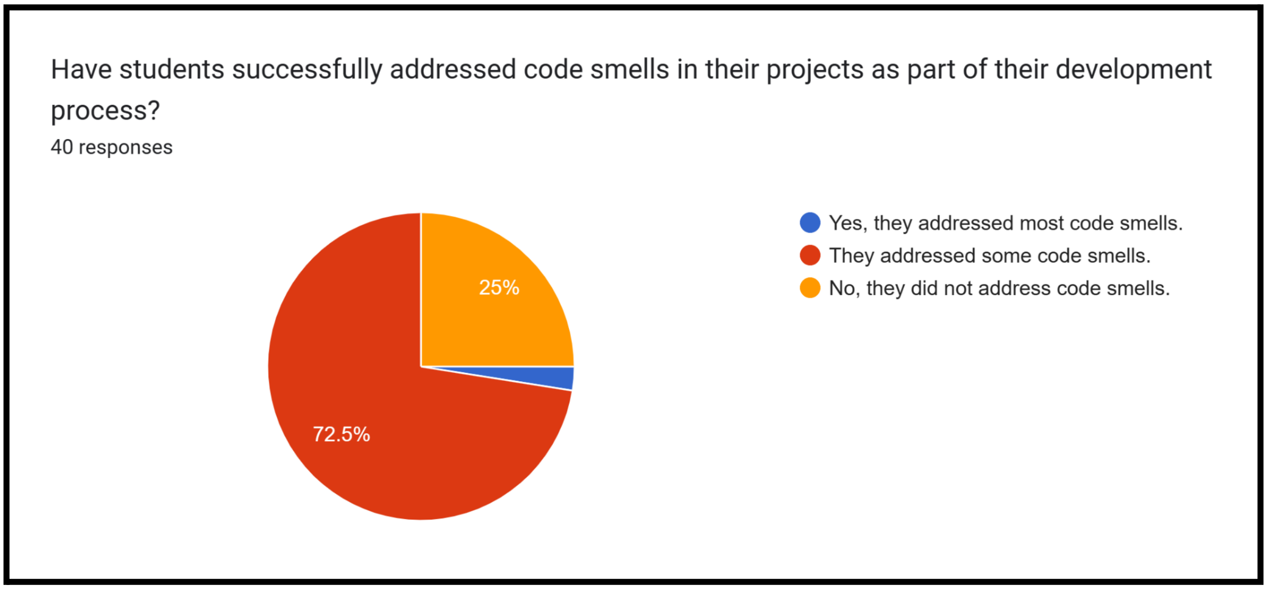


How frequently do students encounter code smells in their projects?

40 responses

Figure 4: Respondents' Frequency of Encountering Code Smells in their Project

Q5

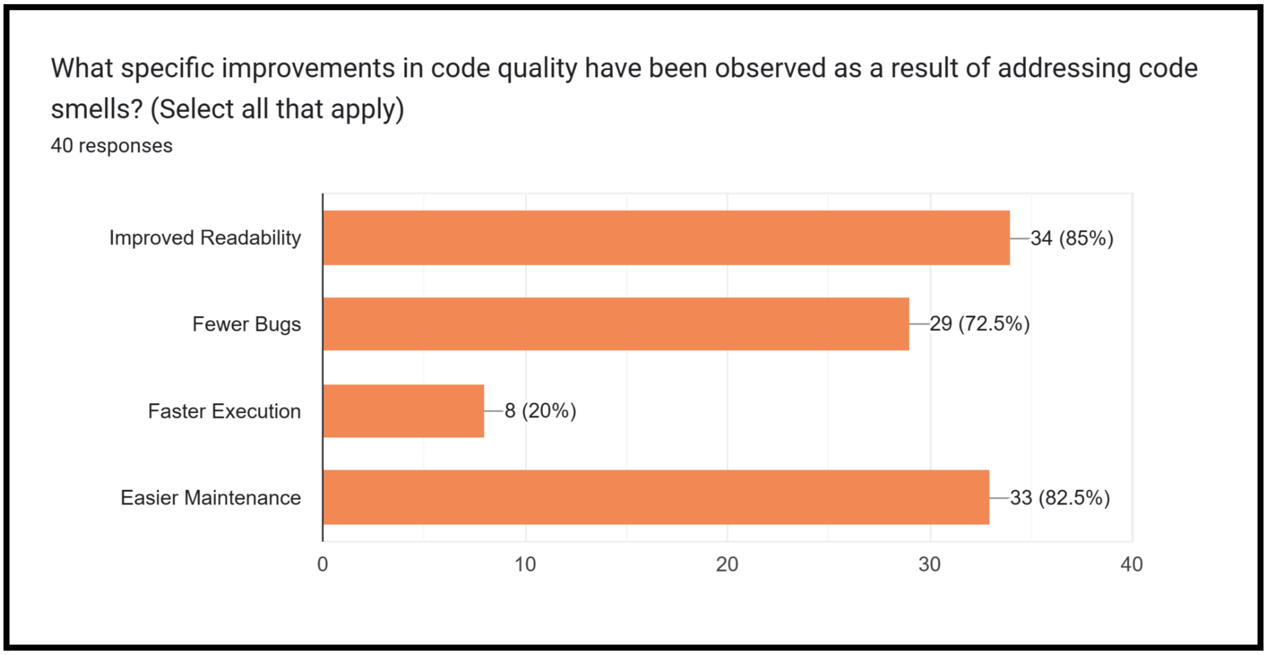


Have students successfully addressed code smells in their projects as part of their development process?

40 responses

Figure 5: Respondents' Opinions about Addressing Code Smells in their project

Q6

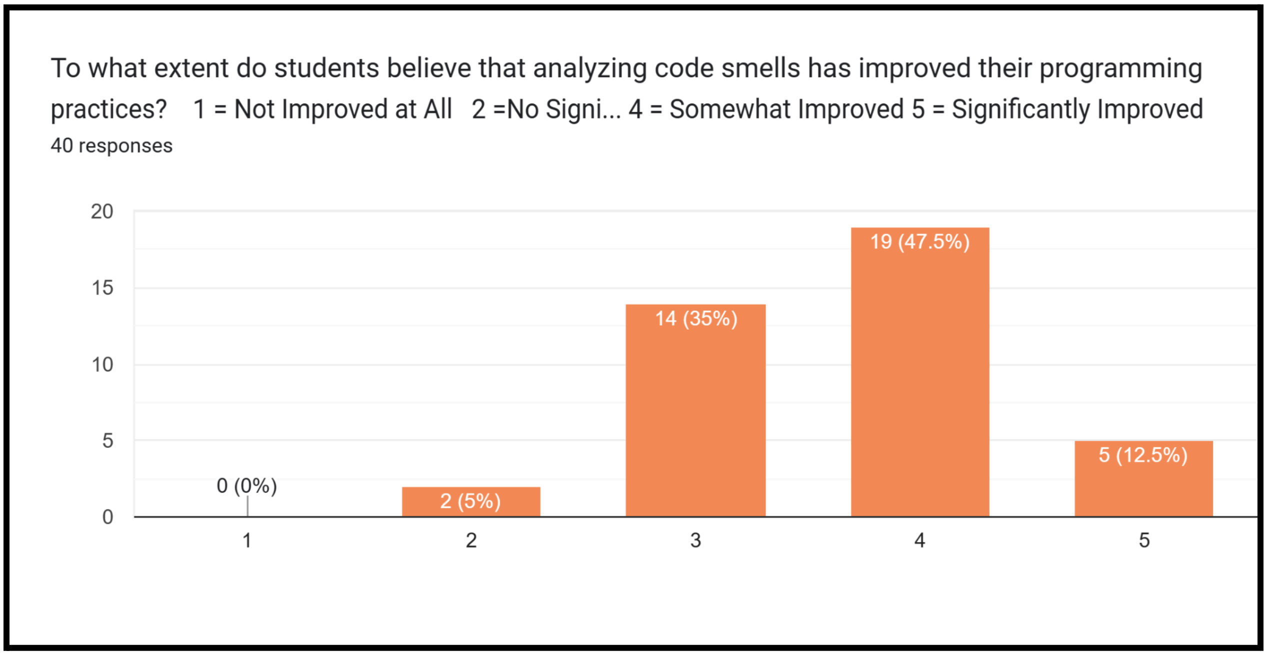


What specific improvements in code quality have been observed as a result of addressing code smells? (Select all that apply)

40 responses

Figure 6: Respondents' Opinions about Improvements in Code Quality as they Addressed Code Smells

Q7

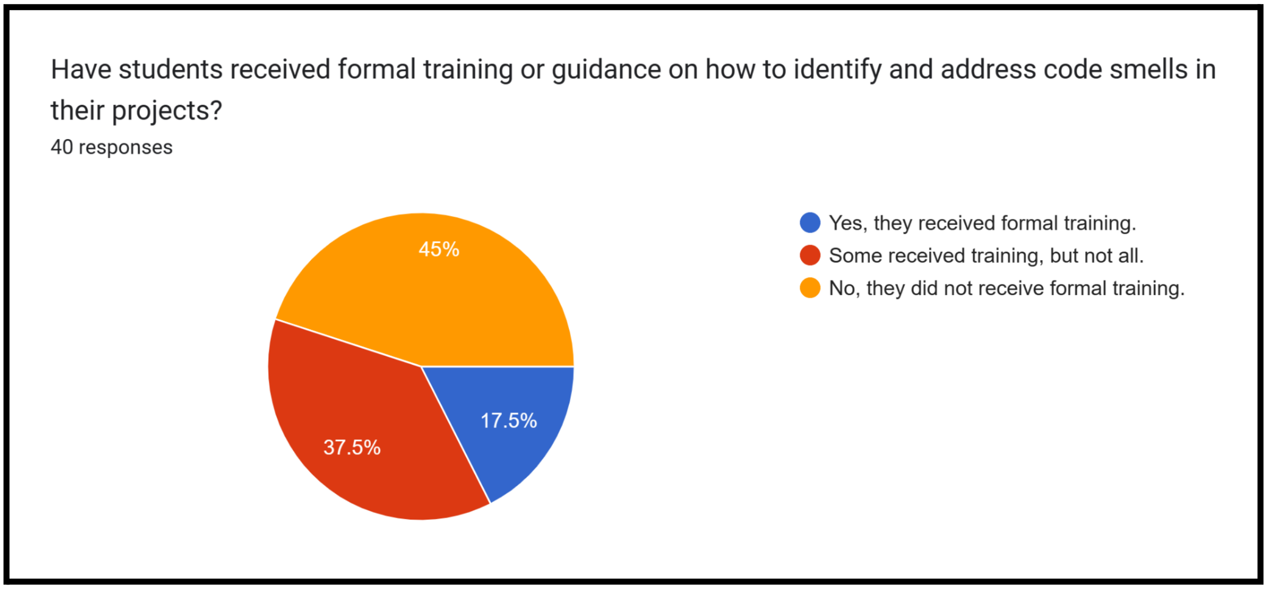


To what extent do students believe that analyzing code smells has improved their programming practices?   1 = Not Improved at All 2 =No Signi... 4 = Somewhat Improved 5 = Significantly Improved

40 responses

Figure 7: Respondents' Belief Analyzing Code Smells to Improve Programming Practices

Q8

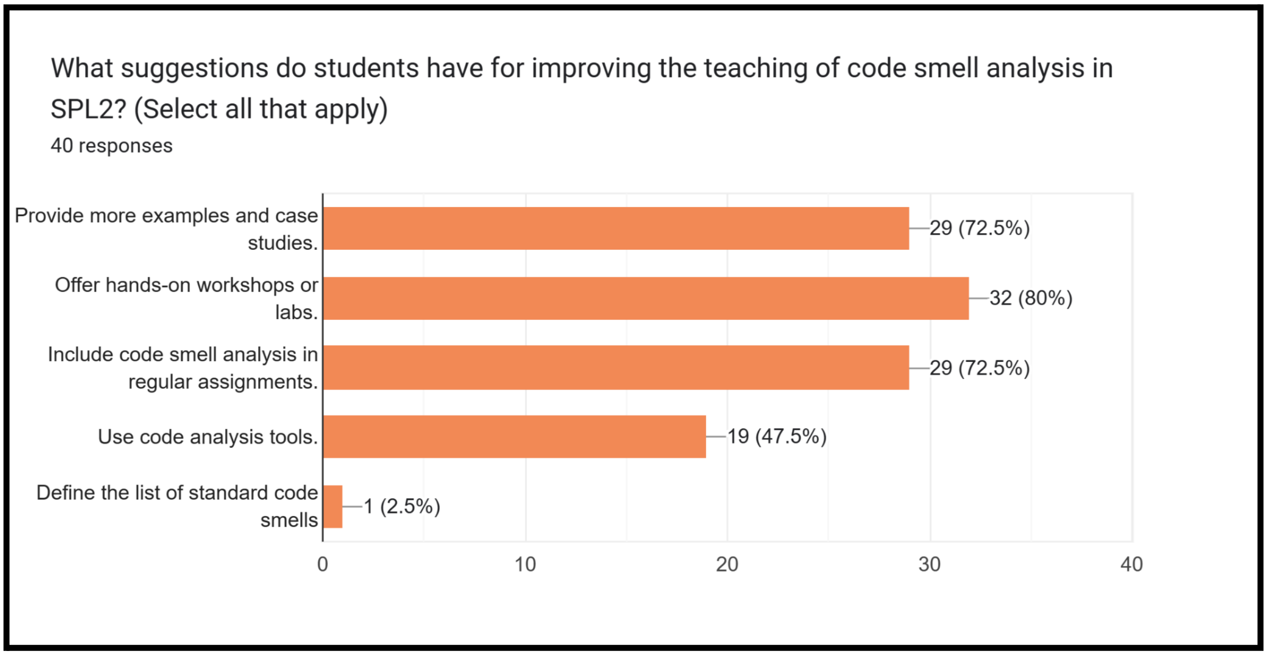


Have students received formal training or guidance on how to identify and address code smells in their projects?

40 responses

Figure 8: Respondents' Opinion on Receiving Formal Training on Identifying & Addressing Code Smells

Q9

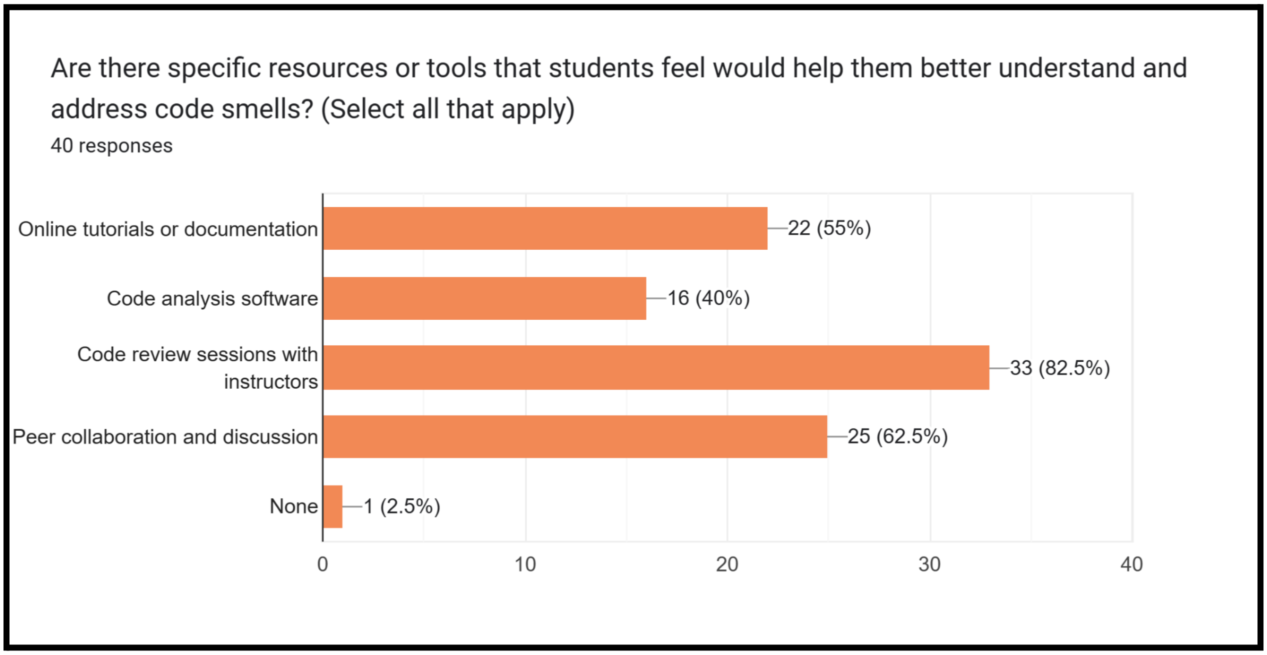


What suggestions do students have for improving the teaching of code smell analysis in SPL2? (Select all that apply)

40 responses

Figure 9: Respondents' Suggestions for Improving the Teaching of Code Smells

Q10

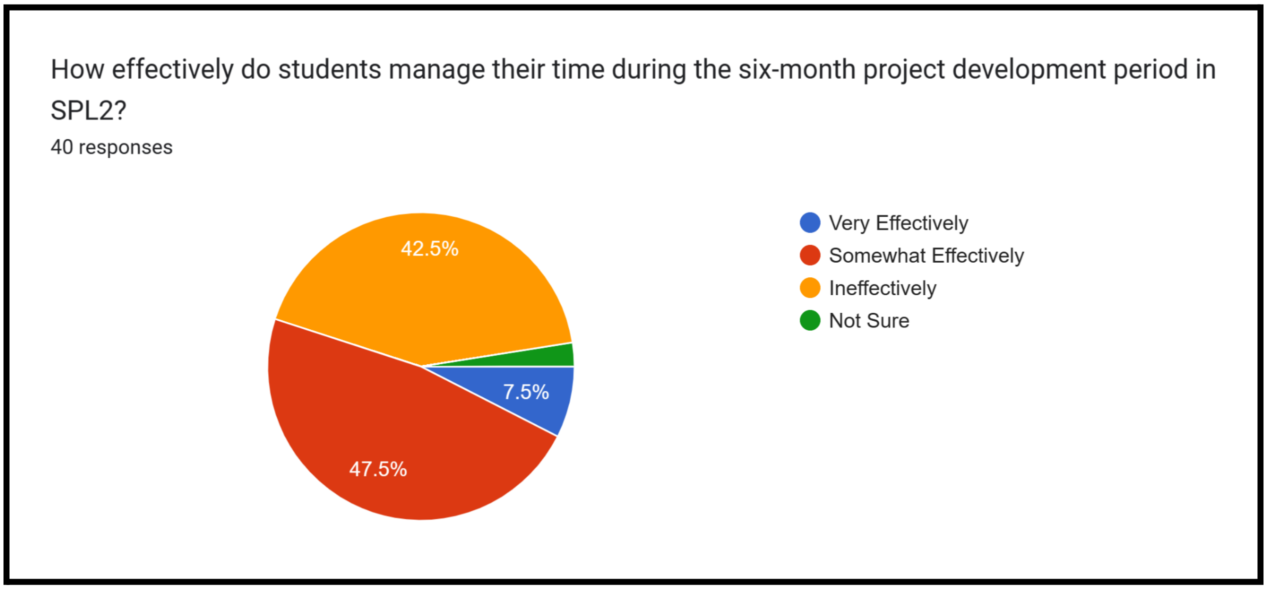


Are there specific resources or tools that students feel would help them better understand and address code smells? (Select all that apply)

40 responses

Figure 10: Respondents' Suggestions on Tools & Resources to Understand & Addressing Code Smells Better

Q11

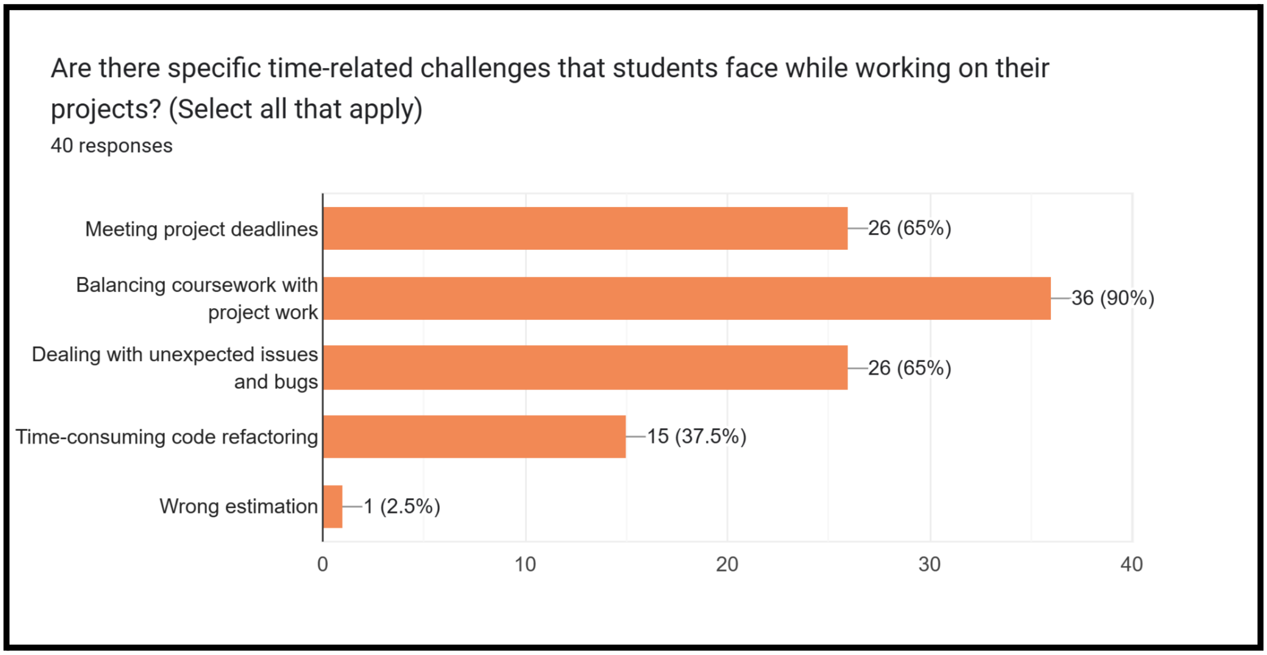


How effectively do students manage their time during the six-month project development period in SPL2?

40 responses

Figure 11: Respondents' Opinion on Managing Time During SPL2

Q12



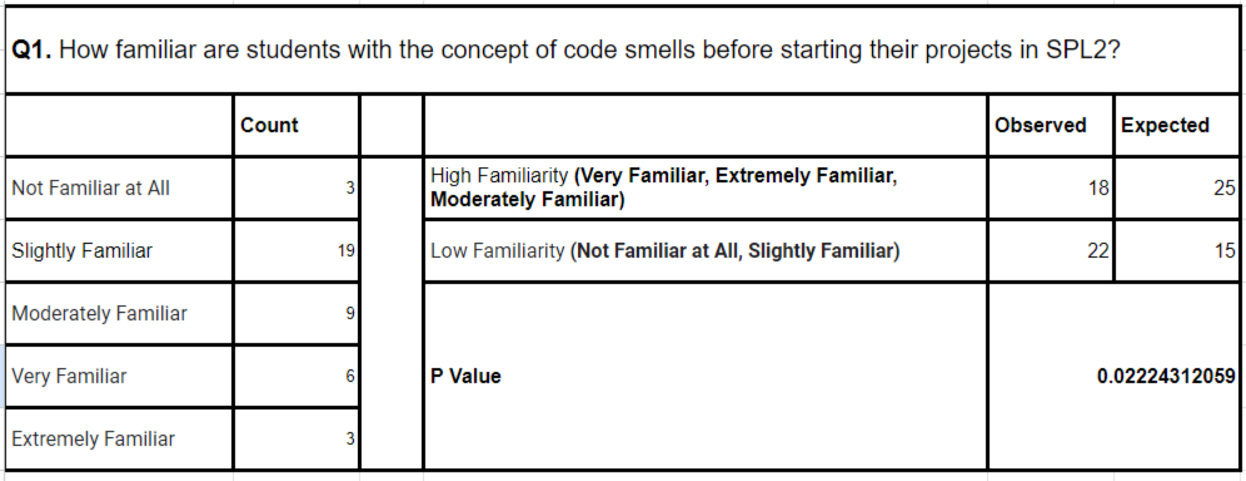
Are there specific time-related challenges that students face while working on their projects? (Select all that apply)

40 responses

Figure 12: Challenges that Respondents Faced while Working on their Project in SPL2

6. Metrics Analysis

Q1. How familiar are students with the concept of code smells before starting their projects in SPL2?



Q1. How familiar are students with the concept of code smells before starting their projects in SPL2?

	Count			Observed	Expected
Not Familiar at All	3		High Familiarity **(Very Familiar, Extremely Familiar, Moderately Familiar)**	18	25
Slightly Familiar	19		Low Familiarity **(Not Familiar at All, Slightly Familiar)**	22	15
Moderately Familiar	9		**P Value**	**0.02224312059**	
Very Familiar	6				
Extremely Familiar	3				

Figure 13: Chi square test on Q1

Students with high familiarity with the concept of code smells before starting their projects in SPL2:	18
Students with low familiarity with the concept of code smells before starting their projects in SPL2:	22

Null Hypothesis (H0): There is no significant difference between the observed and expected counts of students with low familiarity and high familiarity with the concept of code smells before starting their projects in SPL2.

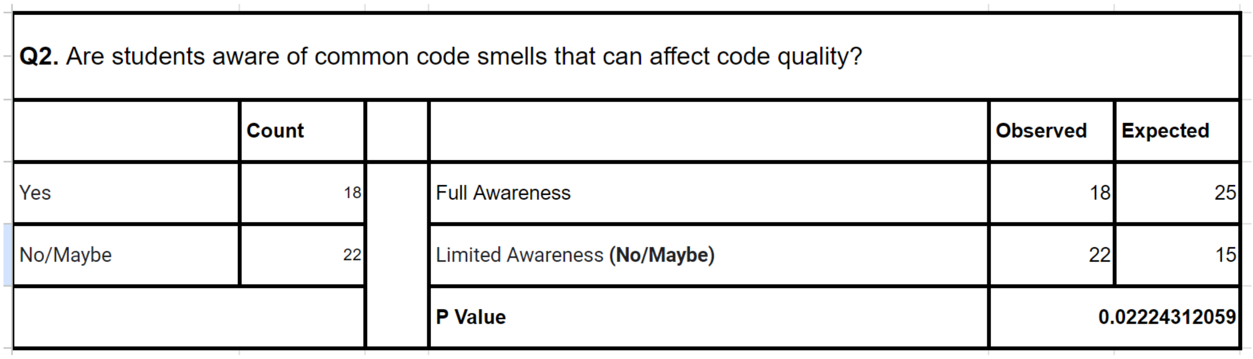
Alternative Hypothesis (Ha): There is a significant difference between the observed and expected counts of students with low familiarity and high familiarity with the concept of code smells before starting their projects in SPL2.

So for category **High Familiarity** our expected value is **25** & for category **Low Familiarity** our expected value is **15**. **Chi square test** was done on the date. The p-value is calculated using the **CHITEST** function. The significance level is considered to be 0.05.

$$P - value = 0.02224312059 < 0.05$$

Decision: Since the p-value **(0.02224312059)** is less than the significance level (0.05), we can reject the null hypothesis. We have enough evidence to conclude that there is a significant difference between the observed and expected counts of students' familiarity with the concept of code smells before starting their projects in SPL2. Therefore, we can say that Students have low familiarity with the concept of code smells before starting their projects in SPL2.

Q2. Are students aware of common code smells that can affect code quality?



Q2. Are students aware of common code smells that can affect code quality?

	Count			**Observed**	**Expected**
Yes	18		Full Awareness	18	25
No/Maybe	22		Limited Awareness **(No/Maybe)**	22	15
			P Value	**0.02224312059**	

Figure 14: Chi square test on Q2

Number of students who possess full awareness of common code smells that affect code quality during SPL2:	18
Number of students who possess limited awareness of common code smells that affect code quality during SPL2:	22

Null Hypothesis (H0): There is no significant difference in students' awareness of common code smells that can affect code quality.

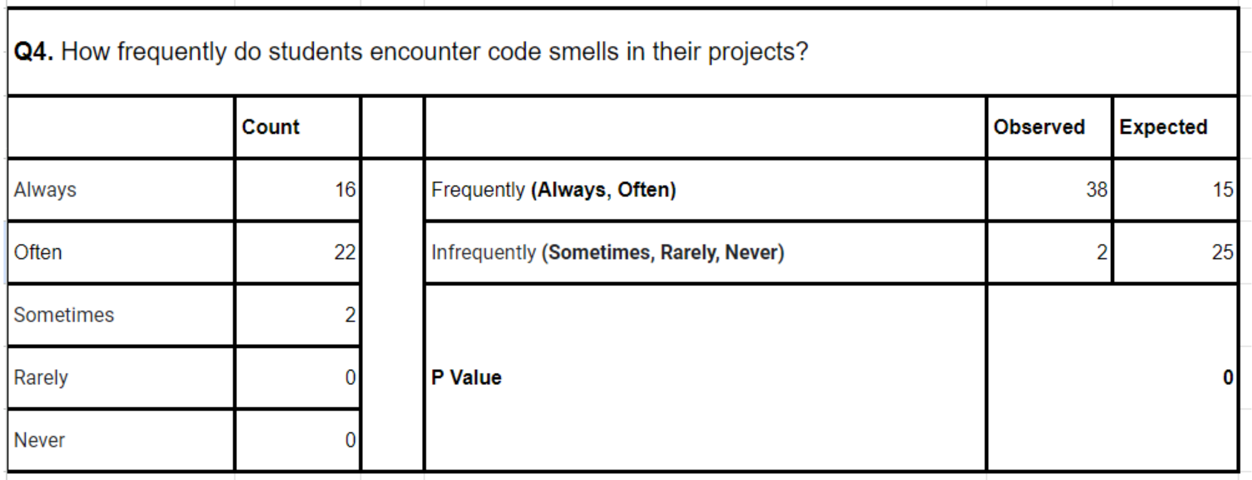
Alternative Hypothesis (Ha): There is a significant difference in students' awareness of common code smells that can affect code quality.

So for category **Full Awareness** our expected value is **25** & for category **Limited Awareness** our expected value is **15**. **Chi square test** was done on the date. The p-value is calculated using the **CHITEST** function. The significance level is considered to be 0.05.

$$P - value = 0.02224312059 < 0.05$$

Decision: Since the p-value **(0.02224312059)** is less than the significance level (0.05), we can reject the null hypothesis. We have enough evidence to conclude that There is a significant difference in students' awareness of common code smells that can affect code quality. Therefore, we can say that Students possess limited awareness of common code smells that affect code quality during SPL2.

Q4. How frequently do students encounter code smells in their projects?



Q4. How frequently do students encounter code smells in their projects?

	Count			Observed	Expected
Always	16		Frequently **(Always, Often)**	38	15
Often	22		Infrequently **(Sometimes, Rarely, Never)**	2	25
Sometimes	2		**P Value**		**0**
Rarely	0				
Never	0				

Figure 15: Chi square test on Q4

Number of students who frequently encounter code smells in their project:	38
Number of students who do not encounter code smells frequently in their project:	2

Null Hypothesis (H0): The distribution of student responses to the frequency of encountering code smells in their projects is uniform and not significantly different from an expected random distribution.

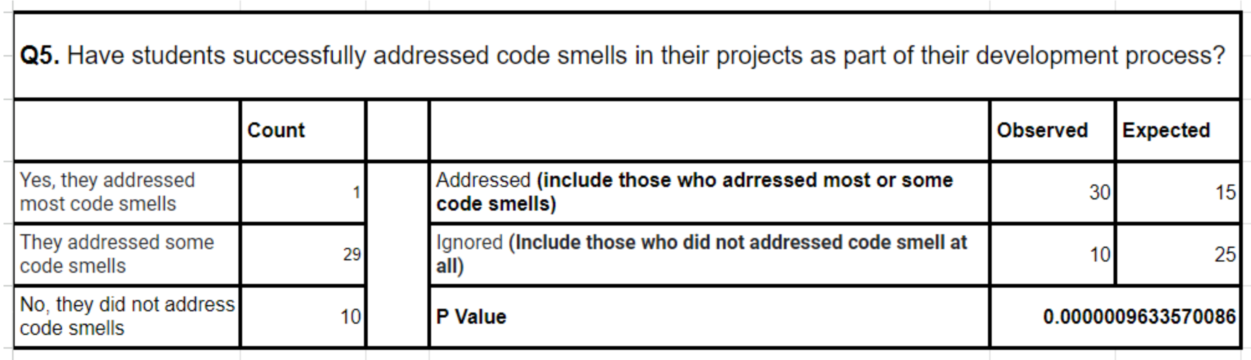
Alternative Hypothesis (Ha): The distribution of student responses to the frequency of encountering code smells in their projects is not uniform, indicating that there is a significant difference in the frequencies reported.

So for category **Frequently** our expected value is **15** & for category **Infrequently** our expected value is **25**. **Chi square test** was done on the date. The p-value is calculated using the **CHITEST** function. The significance level is considered to be 0.05.

$$P - value \ = \ 0.00000000537704 < \ 0.05$$

Decision: Since the p-value **(0.00000000537704)** is less than the significance level (0.05), we can reject the null hypothesis. We have enough evidence to conclude that The distribution of student responses to the frequency of encountering code smells in their projects is not uniform, indicating that there is a significant difference in the frequencies reported. Therefore, we can say that Students frequently encounter code smells in their project during SPL2.

Q5. Have students successfully addressed code smells in their projects as part of their development process?



Q5. Have students successfully addressed code smells in their projects as part of their development process?

	Count			**Observed**	**Expected**
Yes, they addressed most code smells	1		Addressed **(include those who adrressed most or some code smells)**	30	15
They addressed some code smells	29		Ignored **(Include those who did not addressed code smell at all)**	10	25
No, they did not address code smells	10		**P Value**	**0.0000009633570086**	

Figure 16: Chi square test on Q5

Number of students who addressed code smells:	30
Number of students who did not address code smells:	10

***Null Hypothesis (H0)**: The distribution of students' success in addressing code smells is as expected (no significant difference).*

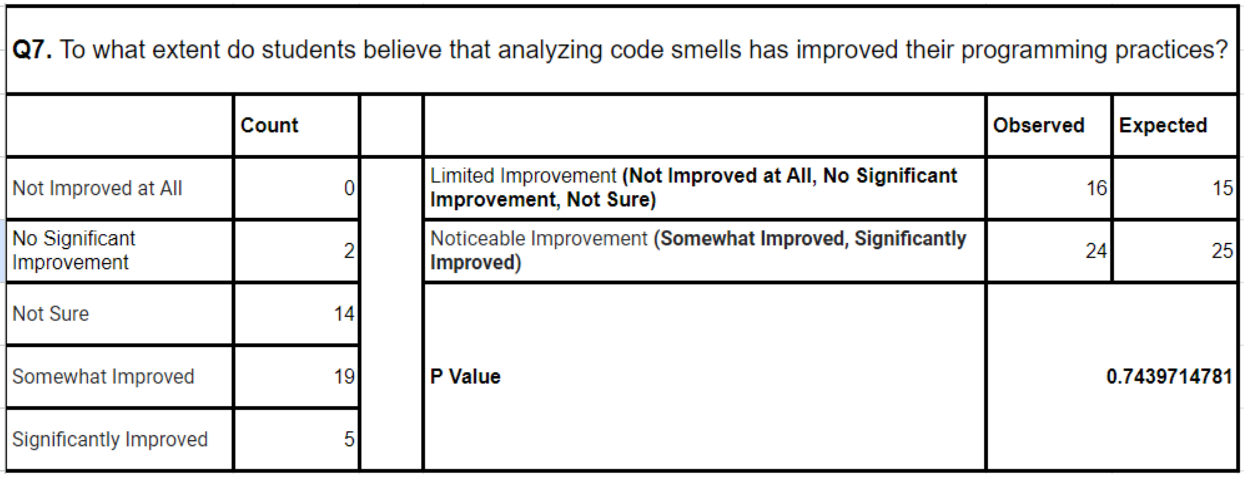
***Alternative Hypothesis (Ha)**: There is a significant difference in the distribution of students' success in addressing code smells.*

So for category **Ignored** our expected value is **25** & for category **Addressed** our expected value is **15**. **Chi square test** was done on the date. The p-value is calculated using the **CHITEST** function. The significance level is considered to be 0.05.

$$P - value = 0.0000009633570086 < 0.05$$

Decision: Since the p-value **(0.0000009633570086)** is less than the significance level (0.05), we can reject the null hypothesis. We have enough evidence to conclude that There is a significant difference in the distribution of students' success in addressing code smells. Therefore, we can say that Students addressed code smells in their project in SPL2.

Q7. To what extent do students believe that analyzing code smells has improved their programming practices?



Q7. To what extent do students believe that analyzing code smells has improved their programming practices?

	Count			Observed	Expected
Not Improved at All	0		Limited Improvement **(Not Improved at All, No Significant Improvement, Not Sure)**	16	15
No Significant Improvement	2		Noticeable Improvement **(Somewhat Improved, Significantly Improved)**	24	25
Not Sure	14		**P Value**	**0.7439714781**	
Somewhat Improved	19				
Significantly Improved	5				

Figure 17: Chi square test on Q7

Number of students who believe that there was limited improvement in their programming practices for analyzing code smell:	16
Number of students who believe that there was noticeable improvement in their programming practices for analyzing code smells:	24

Null Hypothesis (H0): *The distribution of students' perceptions of improvement due to analyzing code smells is as expected (no significant difference).*

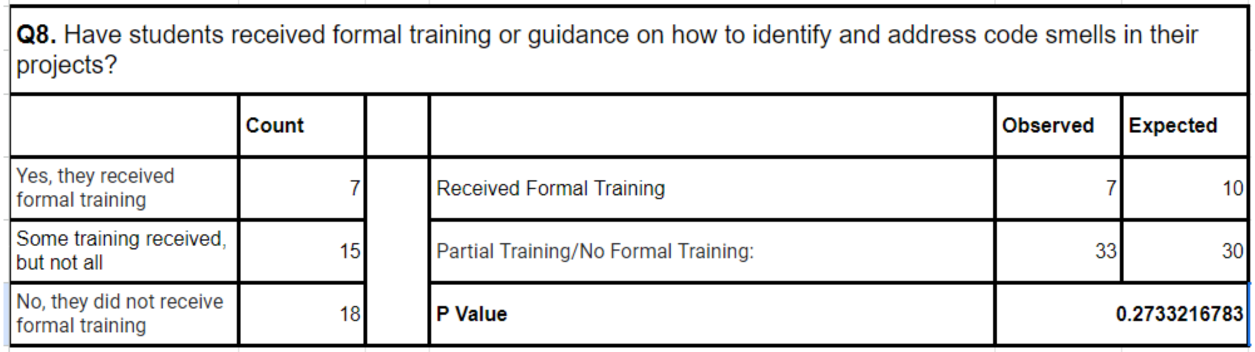
Alternative Hypothesis (Ha): *There is a significant difference in the distribution of students' perceptions of improvement due to analyzing code smells.*

So for category **Noticeable Improvement** our expected value is **25** & for category **Limited Improvement** our expected value is **15**. **Chi square test** was done on the date. The p-value is calculated using the **CHITEST** function. The significance level is considered to be 0.05.

$$P - value = 0.7439714781 > 0.05$$

Decision: Since the p-value **(0.7439714781)** is greater than the significance level (0.05), we fail to reject the null hypothesis. We do not have enough evidence to conclude that There is a significant difference in the distribution of students' perceptions of improvement due to analyzing code smells. Therefore, we can say that Students believe that there was noticeable improvement in their programming practices for analyzing code smells.

Q8. Have students received formal training or guidance on how to identify and address code smells in their projects?



Q8. Have students received formal training or guidance on how to identify and address code smells in their projects?

	Count			Observed	Expected
Yes, they received formal training	7		Received Formal Training	7	10
Some training received, but not all	15		Partial Training/No Formal Training:	33	30
No, they did not receive formal training	18		**P Value**	**0.2733216783**	

Figure 18: Chi square test on Q8

Number of students who received formal training:	7
Number of students who did not receive or partially received formal training:	33

Null Hypothesis (H0)*: There is no significant difference in the proportion of students who received formal training on how to identify and address code smells in their projects.*

Alternative Hypothesis (Ha): *There is a significant difference in the proportion of students who received formal training on how to identify and address code smells in their projects.*

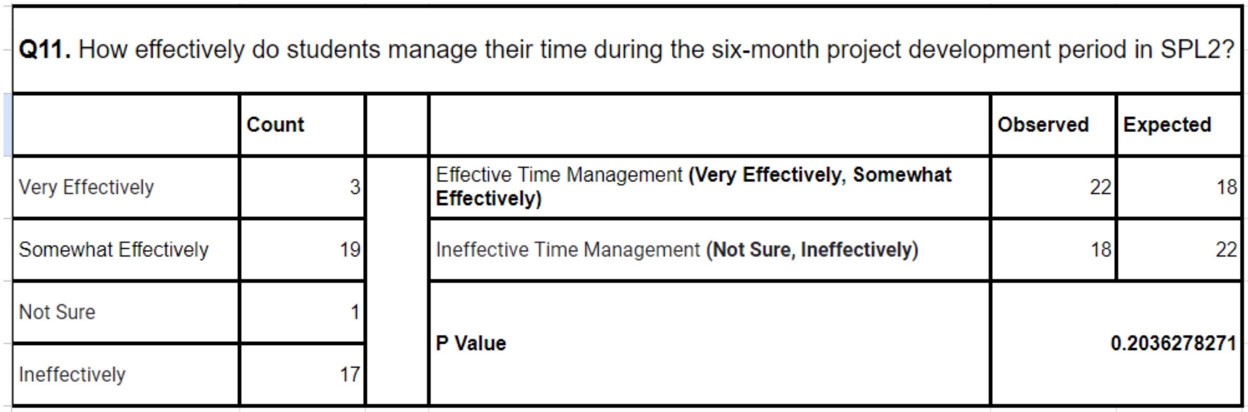
So for both categories our expected value is $\frac{40}{2} = 20$. **Chi square test** was done on the date. The p-value is calculated using the CHITEST function. The significance level is considered to be 0.05.

So for category **Received Formal Training** our expected value is **10** & for category **Partial Training/No Formal Training** our expected value is **30**. **Chi square test** was done on the date. The p-value is calculated using the **CHITEST** function. The significance level is considered to be 0.05.

$$P - value = 0.2733216783 > 0.05$$

Decision: Since the p-value **(0.2733216783)** is greater than the significance level (0.05), we fail to reject the null hypothesis. We do not have enough evidence to conclude that There is a significant difference in the proportion of students who received formal training on how to identify and address code smells. Therefore, we can say that Students did not receive or partially received formal training or guidance on how to identify and address code smells in their projects.

Q11. How effectively do students manage their time during the six-month project development period in SPL2?



Q11. How effectively do students manage their time during the six-month project development period in SPL2?

	Count			**Observed**	**Expected**
Very Effectively	3		Effective Time Management **(Very Effectively, Somewhat Effectively)**	22	18
Somewhat Effectively	19		Ineffective Time Management **(Not Sure, Ineffectively)**	18	22
Not Sure	1		**P Value**	**0.2036278271**	
Ineffectively	17				

Figure 19: Chi square test on Q11

Number of students who could manage time effectively:	22
Number of students who could not manage time effectively:	18

Null Hypothesis (H0): *The distribution of students' ratings for time management is equal across all categories.*

Alternative Hypothesis (Ha): *The distribution of students' ratings for time management is not equal across all categories.*

So for category **Effective Time Management** our expected value is **18** & for category **Ineffective Time Management** our expected value is **22**. **Chi square test** was done on the date. The p-value is calculated using the **CHITEST** function. The significance level is considered to be 0.05.

$$P - value \ = 0.2036278271 > \ 0.05$$

Decision: Since the p-value **(0.2036278271)** is greater than the significance level (0.05), we fail to reject the null hypothesis. We do not have enough evidence to conclude that The distribution of students' ratings for time management is not equal across all categories. Therefore, we can say that Students could not manage their time effectively during the six-month project development period in SPL2.

7. Result of Analysis

We accepted 3 null hypotheses and rejected 4.

SL No.	Questions	Hypotheses	Result
Q1	How familiar are students with the concept of code smells before starting their projects in SPL2?	H_0 : There is no significant difference between the observed and expected counts of students with low familiarity and high familiarity with the concept of code smells before starting their projects in SPL2.	Most of the Students have low familiarity with the concept of code smells before starting their projects in SPL2.
Q2	Are students aware of common code smells that can affect code quality?	H_0 : There is no significant difference in students' awareness of common code smells that can affect code quality	Most of the students possess limited awareness of common code smells that affect code quality during SPL2.
Q4	How frequently do students encounter code smells in their projects?	H_0 : The distribution of student responses to the frequency of encountering code smells in their projects is uniform and not significantly different from an expected random distribution.	Most of the students frequently encounter code smells in their project during SPL2.
Q5	Have students successfully addressed code smells in their projects as part of their development process?	H_0 : The distribution of students' success in addressing code smells is as expected (no significant difference).	Most of the students addressed code smells in their project in SPL2.
Q7	To what extent do students believe that analyzing code smells has improved their programming practices?	H_0 : The distribution of students' perceptions of improvement due to analyzing code smells is as expected (no significant difference).	Most of the students believe that there was noticeable improvement in their programming practices for analyzing code smells
Q8	Have students received formal training or guidance on how to identify and address code smells in their projects?	H_0 : There is no significant difference in the proportion of students who received formal training on how to identify and address code smells in their projects.	Most of the students did not receive or partially received formal training or guidance on how to identify and address code smells in their projects
Q11	How effectively do students manage their time during the six-month project development period in SPL2?	H_0: The distribution of students' ratings for time management is equal across all categories.	Most of the students could not manage their time effectively during the six-month project development period in SPL2.

8. Case Study

A subset of student projects from SPL2 was chosen for analysis, representing a diverse range of applications developed within the curriculum.

8.1 Code Smell Identification

The selected projects underwent a thorough code review to identify prevalent code smells. Identified Code Smells

A. **Long Method:** A notable number of projects exhibited lengthy functions, exceeding commonly accepted thresholds for maintainable code.
B. **Code Duplication:** Instances of repetitive code blocks were prevalent across various projects, indicating a lack of code reuse.
C. **Large Class:** Some classes were overly extensive, violating the Single Responsibility Principle and impacting code organization and clarity.
D. **Inconsistent Naming:** Naming conventions were inconsistent in certain projects, affecting code understandability.
E. **Feature Envy:** Some methods excessively accessed attributes of other classes, suggesting a need for better class organization.

8.2 Prioritization and Refactoring

The identified code smells were prioritized based on severity and potential impact on code quality. Subsequently, refactoring efforts were directed at addressing these prioritized code smells to improve code maintainability and readability.

8.3 Recommendations

A. **Code Review Best Practices:** Encourage students to conduct regular code reviews and adhere to established best practices to catch code smells early in the development process.
B. **Integration of Code Smell Analysis:** Integrate automated code smell detection tools into the development workflow to provide real-time feedback to students during the project development phase.
C. **Educational Emphasis:** Stress the importance of adhering to industry-standard coding practices and introduce students to refactoring techniques as a fundamental part of software engineering education.

9. Interview

Our target audience includes students enrolled in SPL2 or any projects, professional and other potential developers. We interviewed some of the potentials to gather information related to our goal and at the end, we summarized their point as follows-

1. Most of the students were familiar with code smells before starting their projects, they claimed that they just got to know a little bit about code smells through classroom discussions and self research in programming practices. According to them, the most common code smells are commenting in the code, duplicate code, large class and long method.
2. While asking if they could identify the code smells in their code during their projects, one of the interviewees said, "Throughout the project, I often encountered code smells. It had a notable impact on code readability. Mostly whenever I start over the project having a break of days, I used to identify duplicate codes which made my project size larger resulting in large class or long method. I had to change the code for that".
3. Some of the interviewees made an intention to refactor code smells; but most of them did not intend to do it, as a result, they encountered bug issues, slower execution than usual and tough maintenance. Those who refactored their code, claimed that they had improved code readability, reduced complexity and enhanced maintainability. This, in turn, facilitated easier debugging and reduced the likelihood of introducing new bugs during the development process. Ultimately, the code became more robust and efficient, aligning better with established software engineering standards.
4. While asking the most significant challenges in refactoring, one of the interviewees said, "Managing the balance between refactoring and not disrupting the existing functionality". Refactoring too aggressively could potentially introduce new issues or alter the intended behavior. Another interviewee said, "Since this was a team project, effective communication and collaboration among team members were crucial in coordinating the refactoring efforts and addressing challenges promptly.
5. To improve analyzing code smells, students said that more real life examples and workshops with projects on code smells would be beneficial. Introducing them to the tools that would identify code smells like findBugs, sonarQube would be more effective in their learning process.
6. The time constraint of six months posed a challenge in conducting code smells in a full project. It influenced a focused analysis, prioritizing critical code smells for efficient use of limited time. According to the interview, while they were facing a limited timeframe problem in completing the full project within this period, code smells were the least

concern to them. Still those refactored their code, phased an analysis, prioritizing different code smells and refactored critical code smells

10. Conclusion

In conclusion, this project aimed to improve student programming practices in SPL2 by applying Smell Analysis. It set clear goals, assessed code smell awareness, identified common code smells, measured code quality improvements, evaluated the impact, gathered feedback, and assessed time management. Data was collected through a structured questionnaire. The project seeks to enhance code quality, foster a culture of continuous improvement, and contribute to the excellence of software engineering education in the long term.

11. References

- Survey questionnaires link: https://forms.gle/EKnBCmfdSF9EFJov6
- CHI test link: Enhancing Student Programming Practices in SPL2 through Smell A...