An Interview-based Study on Young Developers' Perceptions of Code Smell Detection in Industry

Felipe Ribeiro
Eduardo Fernandes
Eduardo Figueiredo

Summary

1

Introduction

Introduction

- Empirical studies observe that a single software system can have hundreds, even thousands, of code smells. This led researchers and practitioners to employ effort into designing techniques and tools for code smells detection.

Introduction

- Empirical studies observe that a single software system can have hundreds, even thousands, of code smells. This led researchers and practitioners to employ effort into designing techniques and tools for code smells detection.
- Considerable effort has been put on empirically assessing how practitioners perceive code smells as relevant to maintain and evolve software systems.

Introduction

- Empirical studies observe that a single software system can have hundreds, even thousands, of code smells. This led researchers and practitioners to employ effort into designing techniques and tools for code smells detection.
- Considerable effort has been put on empirically assessing how practitioners perceive code smells as relevant to maintain and evolve software systems.
- We particularly advocate that constantly assessing detect code smells in industry is crucial for several reasons. The way developers produce source code evolves rapidly and new technologies emerge.

Introduction

- Empirical studies observe that a single software system can have hundreds, even thousands, of code smells. This led researchers and practitioners to employ effort into designing techniques and tools for code smells detection.
- Considerable effort has been put on empirically assessing how practitioners perceive code smells as relevant to maintain and evolve software systems.
- We particularly advocate that constantly assessing detect code smells in industry is crucial for several reasons. The way developers produce source code evolves rapidly and new technologies emerge.
- This paper introduces an interview-based study on code smell detection in industry.

2

Research questions

Research questions

- **RQ1**: How do young developers define code smells?

Research questions

- **RQ1**: How do young developers define code smells?
  - The traditional literature defines code smells as anomalous code structures that may hinder software maintenance and evolution.

Research questions

- **RQ1**: How do young developers define code smells?
    - The traditional literature defines code smells as anomalous code structures that may hinder software maintenance and evolution.
    - With RQ1, we aim at understanding whether practitioner’s perception on code smells contrasts with the academic wisdom.

Research questions

- **RQ2**: Are young developers concerned about adding code smells to the source code they produce?

Research questions

- **RQ2**: Are young developers concerned about adding code smells to the source code they produce?
  - Several studies suggest that, from the developer perspective, code smells are harmful to software maintenance and evolution.

Research questions

- **RQ2**: Are young developers concerned about adding code smells to the source code they produce?
  - Several studies suggest that, from the developer perspective, code smells are harmful to software maintenance and evolution.
  - Similar to previous studies, we want to understand the extent in which young developers care about adding code smells to their source code.

Research questions

- **RQ2**: Are young developers concerned about adding code smells to the source code they produce?
  - Several studies suggest that, from the developer perspective, code smells are harmful to software maintenance and evolution.
  - Similar to previous studies, we want to understand the extent in which young developers care about adding code smells to their source code.
  - With RQ2, we aim to complement the current knowledge on the concerns of practitioners, given that most previous studies are about ten years old.

Research questions

- **RQ3**: Do young developers use tools to detect code smells on the source code they produce, consume, or maintain?

Research questions

- **RQ3**: Do young developers use tools to detect code smells on the source code they produce, consume, or maintain?
    - Several studies investigate the industry adoption of automated tools for different purposes (refactoring, bug detection, and security assessment).

Research questions

- **RQ3**: Do young developers use tools to detect code smells on the source code they produce, consume, or maintain?
    - Several studies investigate the industry adoption of automated tools for different purposes (refactoring, bug detection, and security assessment).
    - We are aware that, certain developers show reluctance in using tools as they are afraid of side-effects like an expected software quality decay.

Research questions

- **RQ3**: Do young developers use tools to detect code smells on the source code they produce, consume, or maintain?
    - Several studies investigate the industry adoption of automated tools for different purposes (refactoring, bug detection, and security assessment).
    - We are aware that, certain developers show reluctance in using tools as they are afraid of side-effects like an expected software quality decay.
    - With RQ3, we aim at investigating this subject in the context of code smell detection tools.

Research questions

- **RQ3**: Do young developers use tools to detect code smells on the source code they produce, consume, or maintain?
    - Several studies investigate the industry adoption of automated tools for different purposes (refactoring, bug detection, and security assessment).
    - We are aware that, certain developers show reluctance in using tools as they are afraid of side-effects like an expected software quality decay.
    - With RQ3, we aim at investigating this subject in the context of code smell detection tools.
    - We advocate for the use of code smell detection tools because the manual detection can be complex, error-prone, and time consuming.

3

Methodology

Methodology

- The semi structured interview protocol was defined with background and core questions and some possible follow-ups.

Methodology

- The semi structured interview protocol was defined with background and core questions and some possible follow-ups.
- Interviewees that matched the target profile were selected and contacted by convenience from our contact lists.

Methodology

- The semi structured interview protocol was defined with background and core questions and some possible follow-ups.
- Interviewees that matched the target profile were selected and contacted by convenience from our contact lists.
- The interviews were made through Telegram text messages and the answers were then pre-processed.

Methodology

- The semi structured interview protocol was defined with background and core questions and some possible follow-ups.
- Interviewees that matched the target profile were selected and contacted by convenience from our contact lists.
- The interviews were made through Telegram text messages and the answers were then pre-processed.
- A thematic synthesis was employed on the answers, first extracting codes from the tabulated answers (open coding), then building the taxonomies (axial coding).

Methodology

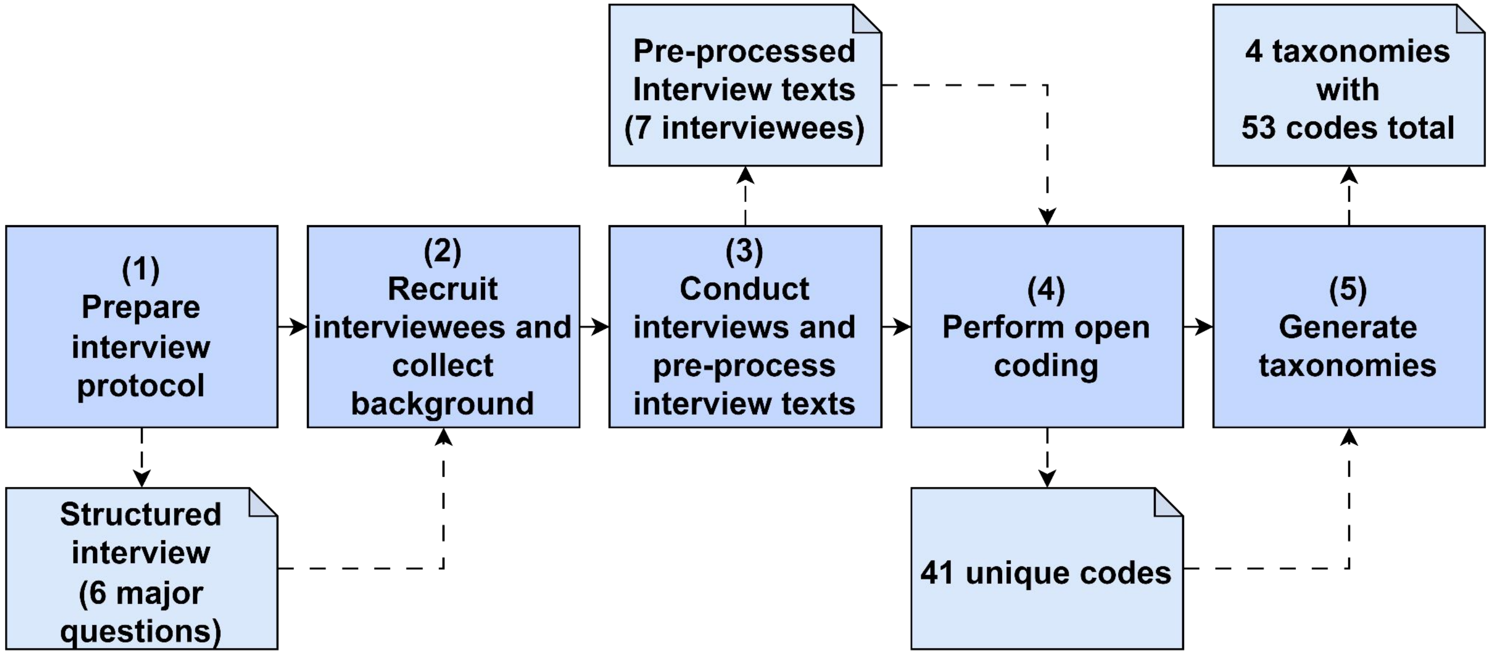
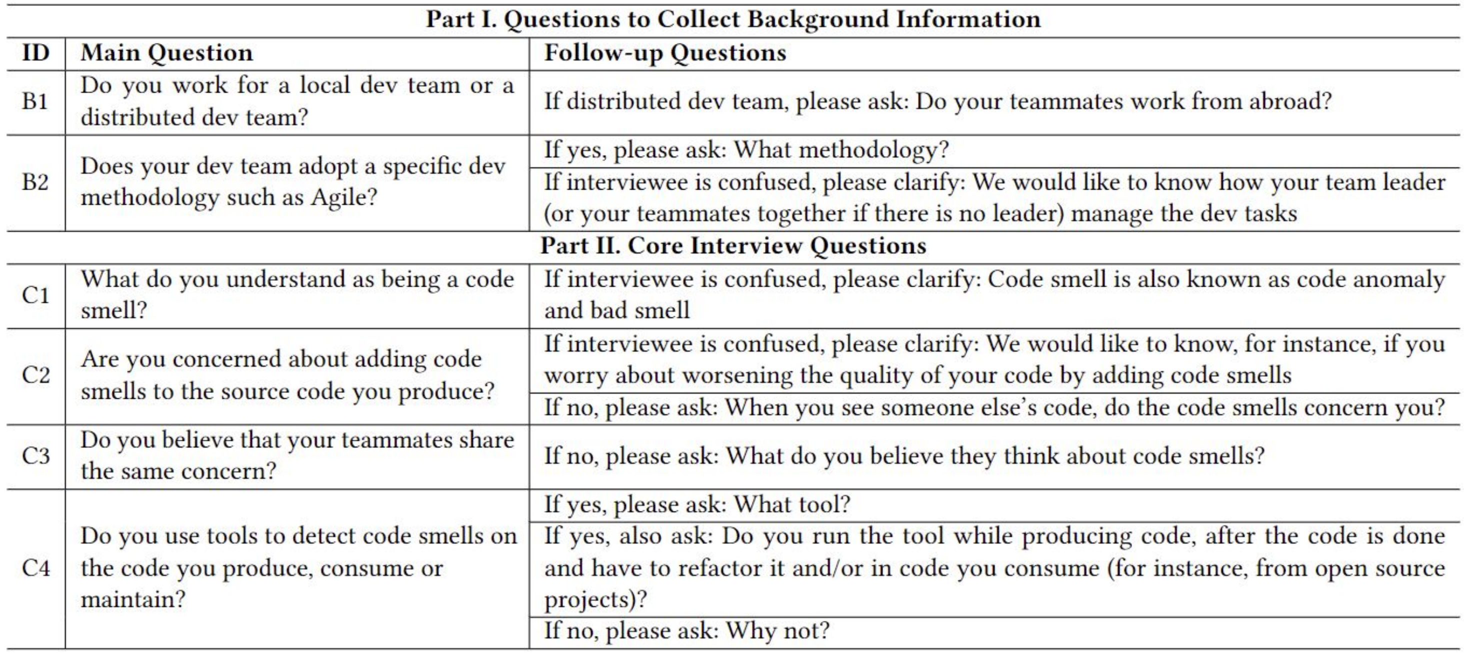

Pre-processed Interview texts (7 interviewees)
4 taxonomies with 53 codes total
(1) Prepare interview protocol
(2) Recruit interviewees and collect background
(3) Conduct interviews and pre-process interview texts
(4) Perform open coding
(5) Generate taxonomies
Structured interview (6 major questions)
41 unique codes

Figure 1: Study Steps

Methodology



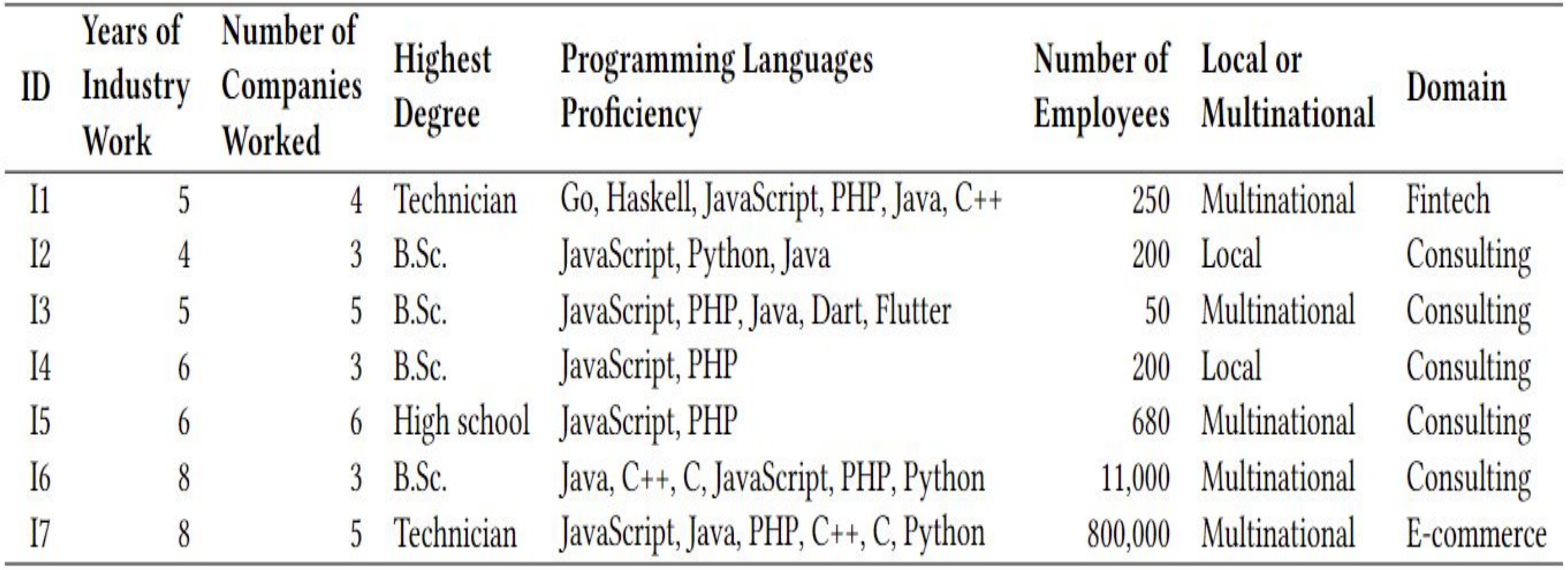
Part I. Questions to Collect Background Information		
ID	**Main Question**	**Follow-up Questions**
B1	Do you work for a local dev team or a distributed dev team?	If distributed dev team, please ask: Do your teammates work from abroad?
B2	Does your dev team adopt a specific dev methodology such as Agile?	If yes, please ask: What methodology?
		If interviewee is confused, please clarify: We would like to know how your team leader (or your teammates together if there is no leader) manage the dev tasks
Part II. Core Interview Questions		
C1	What do you understand as being a code smell?	If interviewee is confused, please clarify: Code smell is also known as code anomaly and bad smell
C2	Are you concerned about adding code smells to the source code you produce?	If interviewee is confused, please clarify: We would like to know, for instance, if you worry about worsening the quality of your code by adding code smells
		If no, please ask: When you see someone else's code, do the code smells concern you?
C3	Do you believe that your teammates share the same concern?	If no, please ask: What do you believe they think about code smells?
C4	Do you use tools to detect code smells on the code you produce, consume or maintain?	If yes, please ask: What tool?
		If yes, also ask: Do you run the tool while producing code, after the code is done and have to refactor it and/or in code you consume (for instance, from open source projects)?
		If no, please ask: Why not?

Table 1: Interview Questions

4

Results

Results - Interviewee Background



ID	Years of Industry Work	Number of Companies Worked	Highest Degree	Programming Languages Proficiency	Number of Employees	Local or Multinational	Domain
I1	5	4	Technician	Go, Haskell, JavaScript, PHP, Java, C++	250	Multinational	Fintech
I2	4	3	B.Sc.	JavaScript, Python, Java	200	Local	Consulting
I3	5	5	B.Sc.	JavaScript, PHP, Java, Dart, Flutter	50	Multinational	Consulting
I4	6	3	B.Sc.	JavaScript, PHP	200	Local	Consulting
I5	6	6	High school	JavaScript, PHP	680	Multinational	Consulting
I6	8	3	B.Sc.	Java, C++, C, JavaScript, PHP, Python	11,000	Multinational	Consulting
I7	8	5	Technician	JavaScript, Java, PHP, C++, C, Python	800,000	Multinational	E-commerce

Table 2: Interviewee Background

Results - RQ1

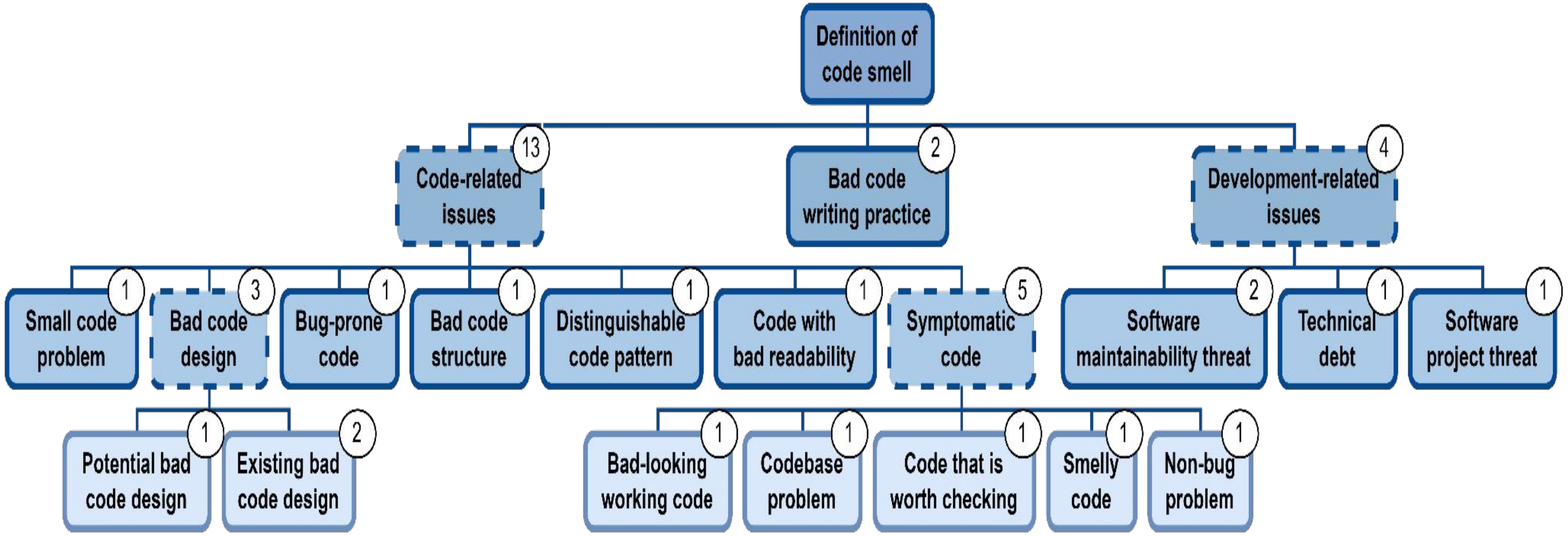


Figure 2: Themes on the Perceptions about Code Smells

Results - RQ1

- There is this assumption in industry that smelly code can lead to bugs, gradually confirmed by previous empirical researches.

Results - RQ1

- There is this assumption in industry that smelly code can lead to bugs, gradually confirmed by previous empirical researches.
- Mentioning that a code smell might be a technical debt, implies a need for refactoring at some point during the life cycle of a software system, showing the code smell relevance at some extent.

Results - RQ1

- There is this assumption in industry that smelly code can lead to bugs, gradually confirmed by previous empirical researches.
- Mentioning that a code smell might be a technical debt, implies a need for refactoring at some point during the life cycle of a software system, showing the code smell relevance at some extent.
- In the end, we noticed that all answers are in line with the traditional definition of code smells, even when some interviewees lacked higher education. This could lead to the perception that the intuition behind code smells might be learned by practice.

Results - RQ2

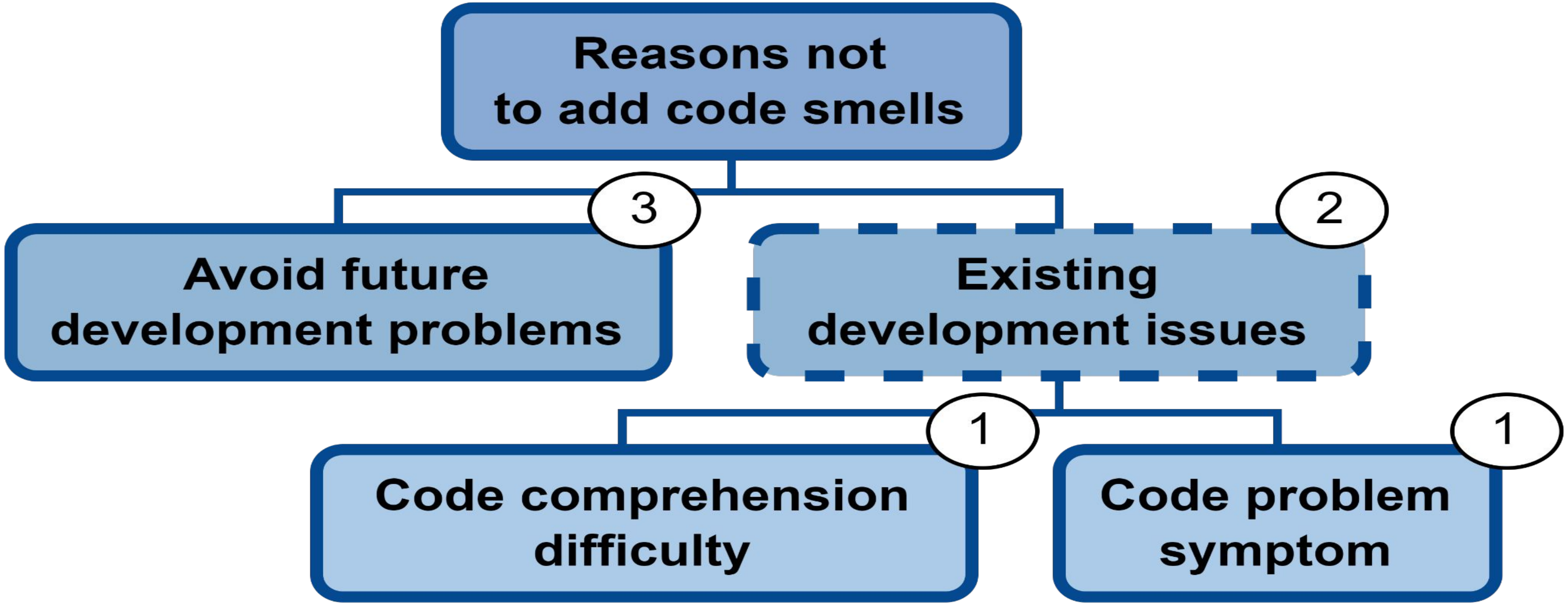

Reasons not to add code smells
3
Avoid future development problems
2
Existing development issues
1
Code comprehension difficulty
1
Code problem symptom

Figure 3: Themes on the Reasons Not to Add Code Smells

Results - RQ2

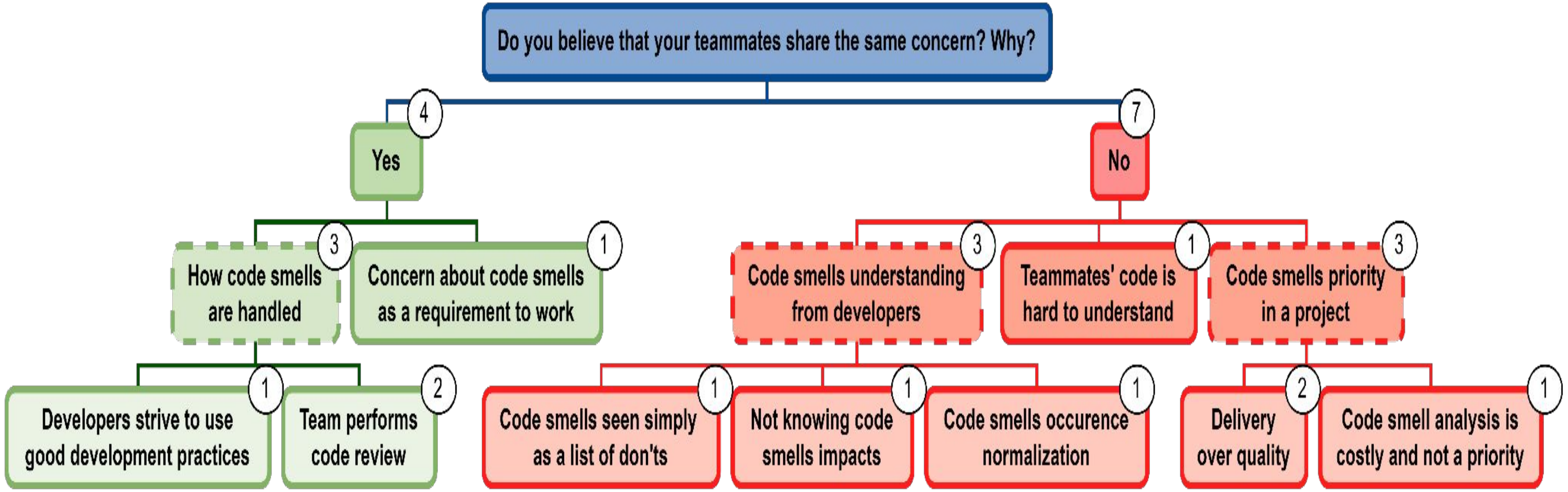

Do you believe that your teammates share the same concern? Why?
4
Yes
7
No
How code smells are handled
3
Concern about code smells as a requirement to work
1
Code smells understanding from developers
3
Teammates' code is hard to understand
1
Code smells priority in a project
3
Developers strive to use good development practices
1
Team performs code review
2
Code smells seen simply as a list of don'ts
1
Not knowing code smells impacts
1
Code smells occurence normalization
1
Delivery over quality
2
Code smell analysis is costly and not a priority
1

Figure 4: Themes on Why Young Developers Believe Their Teammates (Do Not) Share Their Concerns

Results - RQ2

- All the seven interviewees claimed that they are concerned with adding code smells to their source code, even reassuring some reasons why this addition would be an issue.

Results - RQ2

- All the seven interviewees claimed that they are concerned with adding code smells to their source code, even reassuring some reasons why this addition would be an issue.
- However, only a half of them feel that their teammates share the same concern.

Results - RQ2

- All the seven interviewees claimed that they are concerned with adding code smells to their source code, even reassuring some reasons why this addition would be an issue.
- However, only a half of them feel that their teammates share the same concern.
- Raise awareness on the practical relevance of avoiding and eliminating code smells could be a way to support future maintenance and evolution tasks.

Results - RQ3

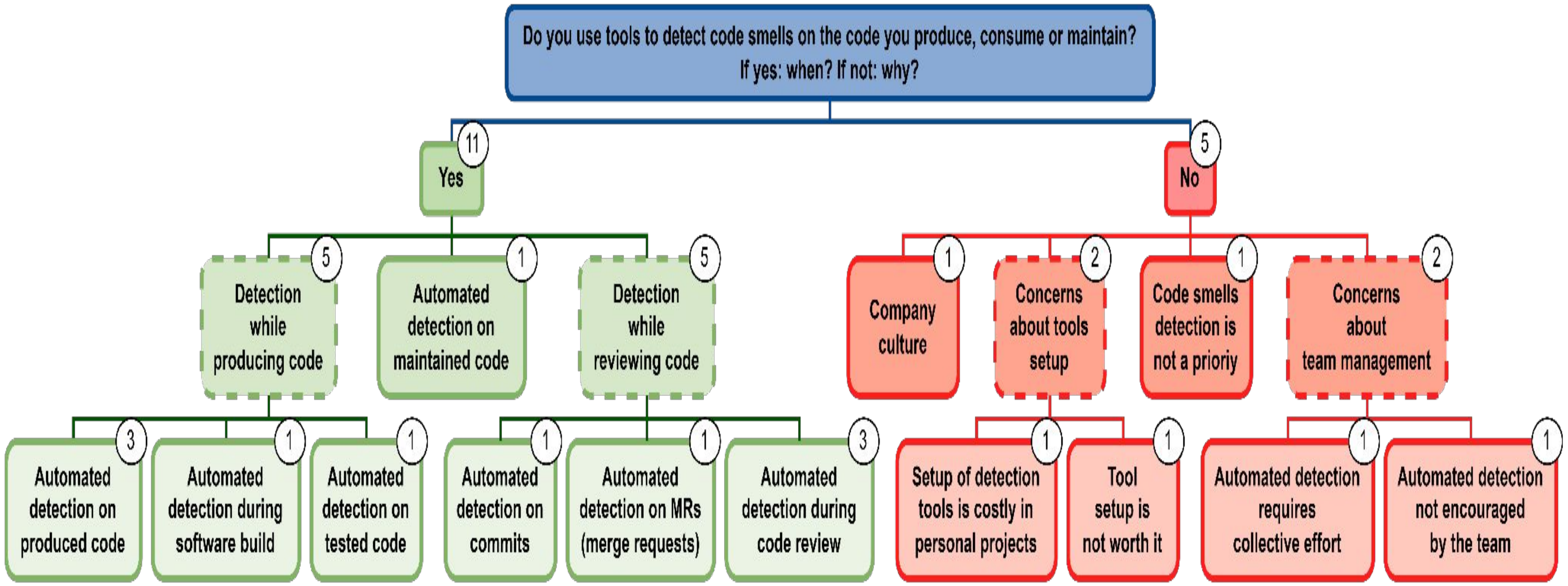

Do you use tools to detect code smells on the code you produce, consume or maintain? If yes: when? If not: why?
Yes 11
No 5
Detection while producing code 5
Automated detection on maintained code 1
Detection while reviewing code 5
Company culture 1
Concerns about tools setup 2
Code smells detection is not a prioriy 1
Concerns about team management 2
Automated detection on produced code 3
Automated detection during software build 1
Automated detection on tested code 1
Automated detection on commits 1
Automated detection on MRs (merge requests) 1
Automated detection during code review 3
Setup of detection tools is costly in personal projects 1
Tool setup is not worth it 1
Automated detection requires collective effort 1
Automated detection not encouraged by the team 1

Figure 5: Do you use tools to detect code smells on the code you produce, consume or maintain? If yes: when? If not: why?

Results - RQ3

- Most interviewees use code smell detection tools and language-specific linters.

Results - RQ3

- Most interviewees use code smell detection tools and language-specific linters.
- Unfortunately, costs associated with tool setup, as well as company culture, may prevent developers from using tools.

Results - RQ3

- Most interviewees use code smell detection tools and language-specific linters.
- Unfortunately, costs associated with tool setup, as well as company culture, may prevent developers from using tools.
- Overall, young developers seem to be willing to use code smell detection tools if properly encouraged.

5

Threats to validity

Threats to validity

- **Internal Validity**: Using Telegram as the tool for the interviews.

Threats to validity

- **Internal Validity**: Using Telegram as the tool for the interviews.
    - This might be an issue due to the possible lack of engagement of the interviewees during the interviews.

Threats to validity

- **Internal Validity**: Using Telegram as the tool for the interviews.
  - This might be an issue due to the possible lack of engagement of the interviewees during the interviews.
  - The tool allowed to closely interact with the interviewees and ask questions in a more efficient and reactive way to mitigate this issue.

Threats to validity

- **Internal Validity**: Using Telegram as the tool for the interviews.
    - This might be an issue due to the possible lack of engagement of the interviewees during the interviews.
    - The tool allowed to closely interact with the interviewees and ask questions in a more efficient and reactive way to mitigate this issue.
- **Conclusion Validity:** Possibility to not analyze the data correctly.

Threats to validity

- **Internal Validity**: Using Telegram as the tool for the interviews.
  - This might be an issue due to the possible lack of engagement of the interviewees during the interviews.
  - The tool allowed to closely interact with the interviewees and ask questions in a more efficient and reactive way to mitigate this issue.
- **Conclusion Validity:** Possibility to not analyze the data correctly.
  - This could result in losing important data analysis or even lower quality conclusions.

Threats to validity

- **Internal Validity**: Using Telegram as the tool for the interviews.
  - This might be an issue due to the possible lack of engagement of the interviewees during the interviews.
  - The tool allowed to closely interact with the interviewees and ask questions in a more efficient and reactive way to mitigate this issue.
- **Conclusion Validity:** Possibility to not analyze the data correctly.
  - This could result in losing important data analysis or even lower quality conclusions.
  - We performed the thematic synthesis based on literature guidelines and during pairing sessions.

Threats to validity

- **External Validity:** Recruiting by convenience and amount of interviewees.

Threats to validity

- **External Validity:** Recruiting by convenience and amount of interviewees.
    - It may be very challenging to generalize our study findings.

Threats to validity

- **External Validity:** Recruiting by convenience and amount of interviewees.
    - It may be very challenging to generalize our study findings.
    - We did the best we could to achieve diversity in the interviewee background.

Threats to validity

- **External Validity:** Recruiting by convenience and amount of interviewees.
  - It may be very challenging to generalize our study findings.
  - We did the best we could to achieve diversity in the interviewee background.
- **Construct Validity:** The construction of the interview process.

Threats to validity

- **External Validity:** Recruiting by convenience and amount of interviewees.
  - It may be very challenging to generalize our study findings.
  - We did the best we could to achieve diversity in the interviewee background.
- **Construct Validity:** The construction of the interview process.
  - A poorly structured interview protocol could lead to interviews that would not answer our research questions.

Threats to validity

- **External Validity:** Recruiting by convenience and amount of interviewees.
  - It may be very challenging to generalize our study findings.
  - We did the best we could to achieve diversity in the interviewee background.
- **Construct Validity:** The construction of the interview process.
  - A poorly structured interview protocol could lead to interviews that would not answer our research questions.
  - We defined the interview protocol in pairs and iteratively.

6

Related works

Related works

- Aiko Yamashita and Leon Moonen. 2012. Do code smells reflect important maintainability aspects?. In Proceedings of the 28th International Conference on Software Maintenance (ICSM). 306–315.

Related works

- Aiko Yamashita and Leon Moonen. 2012. Do code smells reflect important maintainability aspects?. In Proceedings of the 28th International Conference on Software Maintenance (ICSM). 306–315.
- Aiko Yamashita and Leon Moonen. 2013. Do developers care about code smells? An exploratory survey. In Proceedings of the 20th Working Conference on Reverse Engineering (WCRE). 242–251.

Related works

- Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Andrea De Lucia. 2014. Do they really smell bad? A study on developers' perception of bad code smells. In Proceedings of the 30th International Conference on Software Maintenance and Evolution (ICSME). 101–110.

Related works

- Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Andrea De Lucia. 2014. Do they really smell bad? A study on developers' perception of bad code smells. In Proceedings of the 30th International Conference on Software Maintenance and Evolution (ICSME). 101–110.
- Davide Taibi, Andrea Janes, and Valentina Lenarduzzi. 2017. How developers perceive smells in source code: A replicated study. Information and Software Technology (IST) 92 (2017), 223–235.

7

Next steps

Next steps

- Greatly increase the amount of interviewees and their diversity.

Next steps

- Greatly increase the amount of interviewees and their diversity.
- Replicate the interview with undergraduate and graduate students and compare the practitioners’ perception with a more academic-oriented perception.

Next steps

- Greatly increase the amount of interviewees and their diversity.
- Replicate the interview with undergraduate and graduate students and compare the practitioners’ perception with a more academic-oriented perception.
- Replicate the interview with contributors to Open Source Software (OSS) projects.

An Interview-based Study on Young Developers' Perceptions of Code Smell Detection in Industry

Felipe Ribeiro
Eduardo Fernandes
Eduardo Figueiredo