An Interview Study on 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.

Introduction

- Empirical studies observe that a single software system can have hundreds, even thousands, of code smells.
- Considerable effort has been put on assessing how practitioners perceive code smells as relevant to maintain software systems.

2

Research questions

Research questions

- **RQ1**: How do 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 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 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 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 developers care about adding code smells to their source code.

Research questions

- **RQ2**: Are 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 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 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 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 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.

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, LinkedIn and WhatsApp 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, LinkedIn and WhatsApp 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

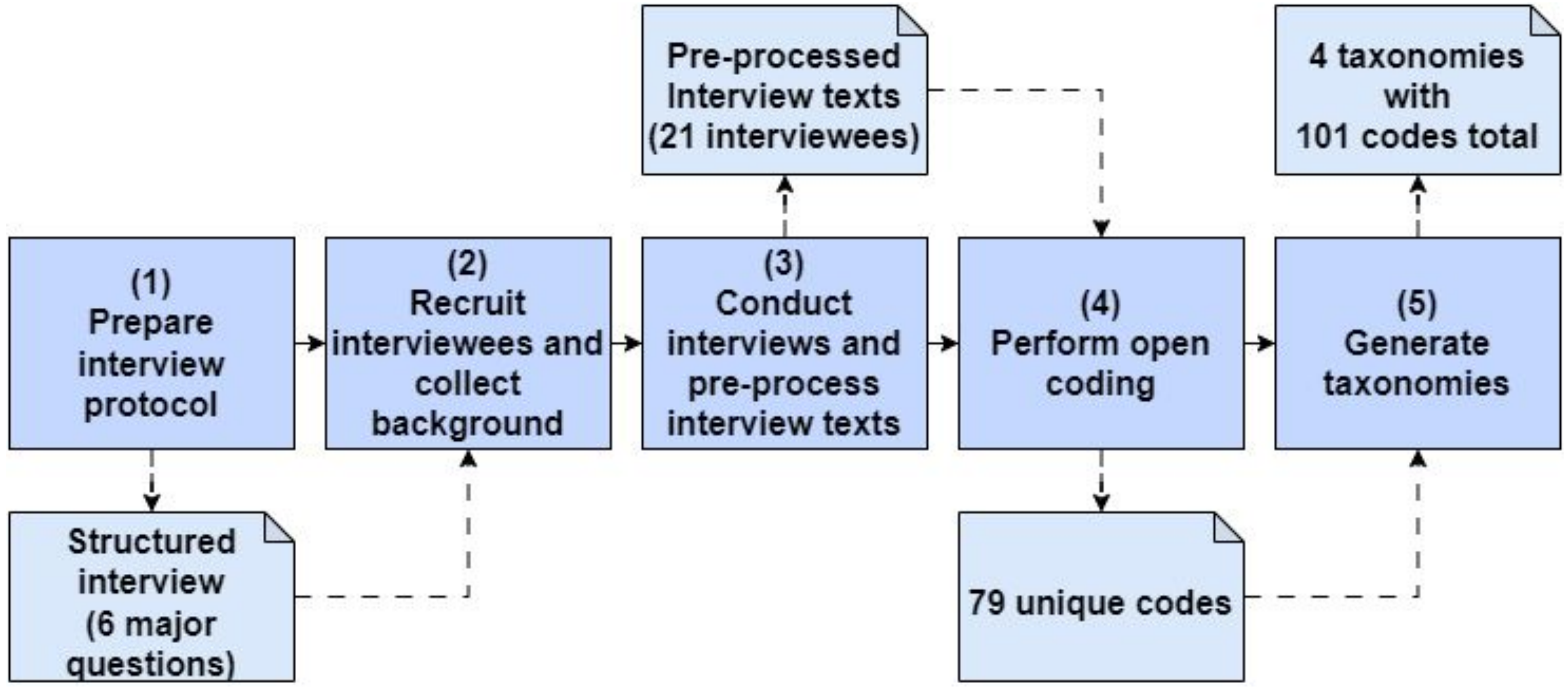
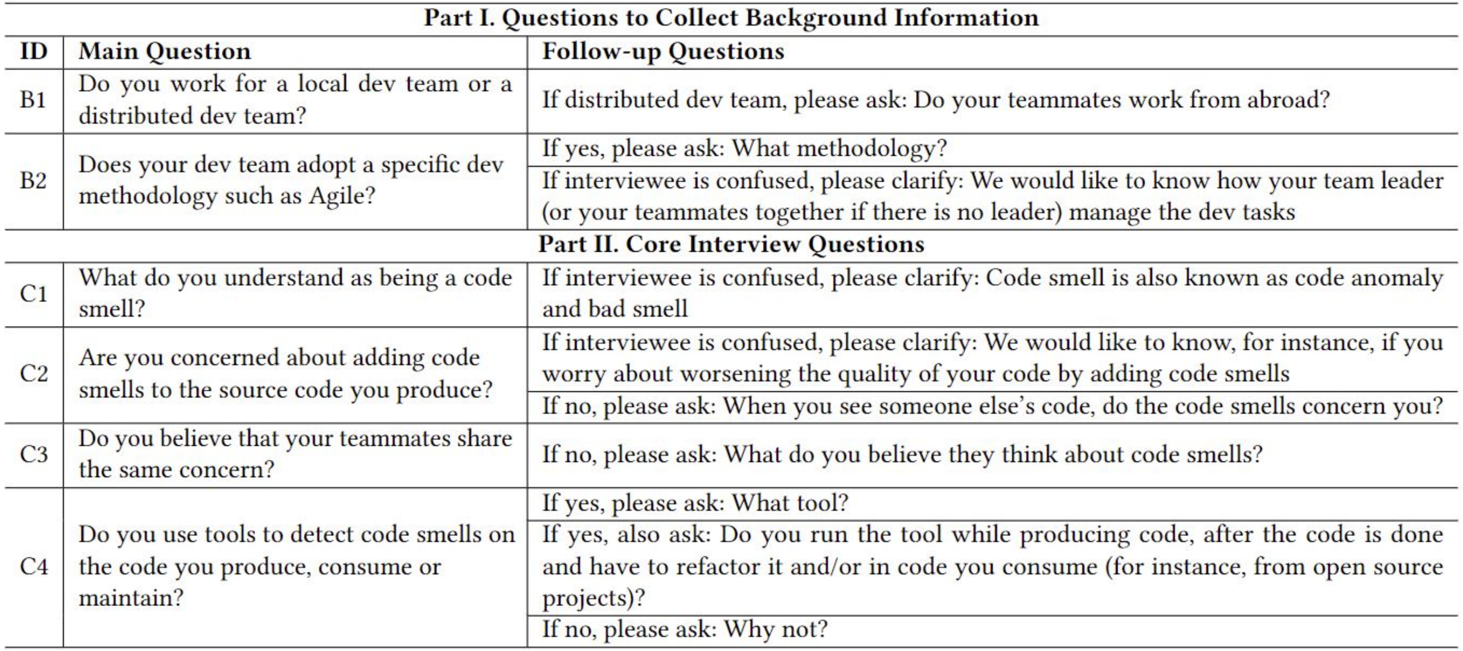

Pre-processed Interview texts (21 interviewees)
4 taxonomies with 101 codes total
(1) Prepare interview protocol
(2) Recruit interviewees and collect background
(3) Conduct interviews and pre-process interview texts
(4) Perform open coding
(5) Generate taxonomies
Structured interview (6 major questions)
79 unique codes

Figure 1: Study Steps

Methodology



ID	Main Question	Follow-up Questions
Part I. Questions to Collect Background Information		
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 - RQ1

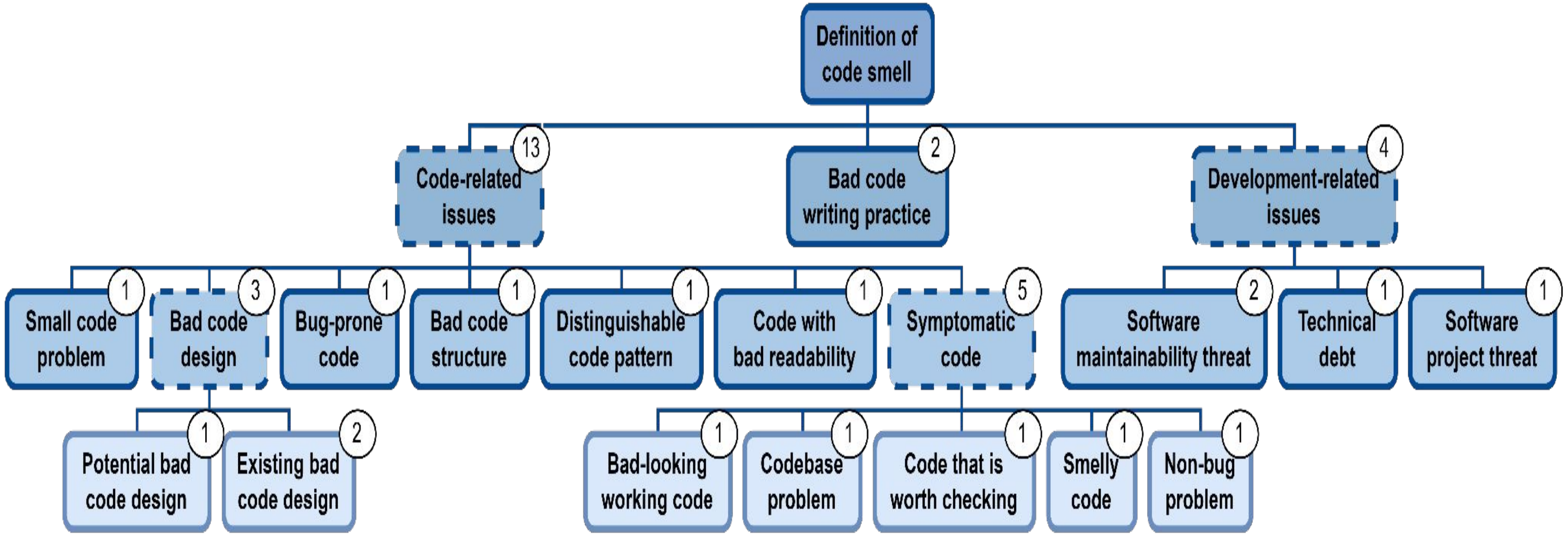

Definition of code smell
Code-related issues 13
Bad code writing practice 2
Development-related issues 4
Small code problem 1
Bad code design 3
Bug-prone code 1
Bad code structure 1
Distinguishable code pattern 1
Code with bad readability 1
Symptomatic code 5
Software maintainability threat 2
Technical debt 1
Software project threat 1
Potential bad code design 1
Existing bad code design 2
Bad-looking working code 1
Codebase problem 1
Code that is worth checking 1
Smelly code 1
Non-bug problem 1

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

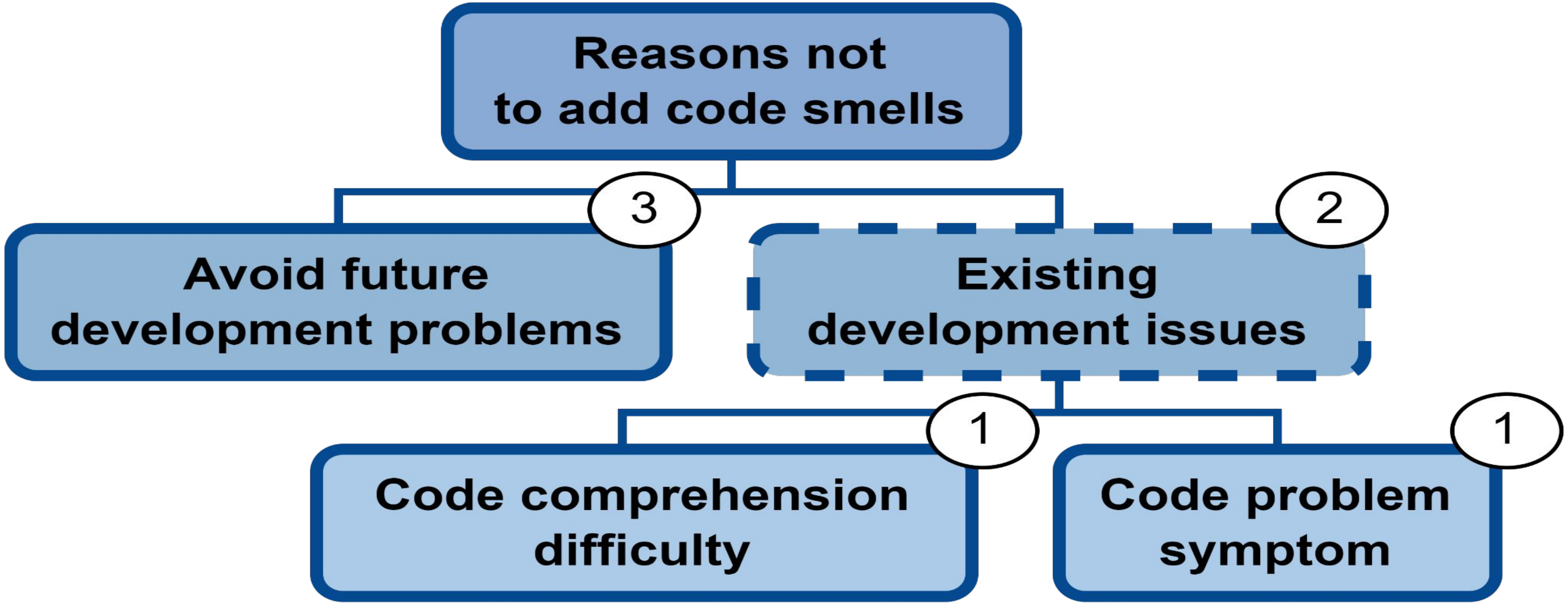

Reasons not to add code smells
3
Avoid future development problems
2
Existing development issues
1
Code comprehension difficulty
1
Code problem symptom

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

Results - RQ2

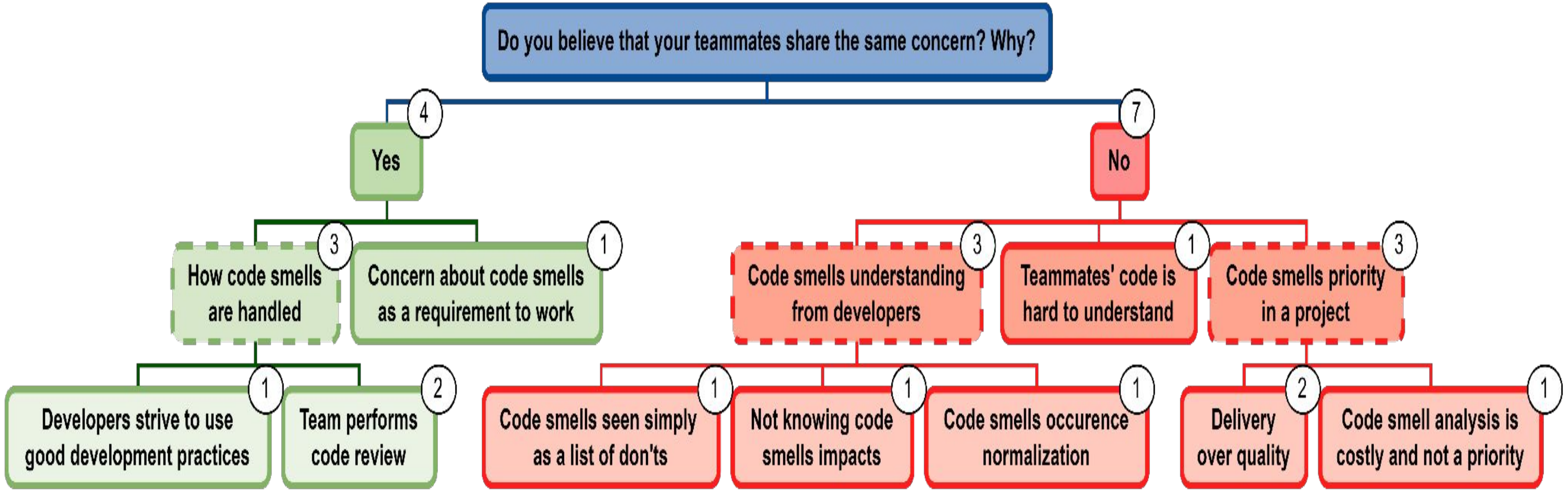

Do you believe that your teammates share the same concern? Why?
4
Yes
7
No
How code smells are handled
3
Concern about code smells as a requirement to work
1
Code smells understanding from developers
3
Teammates' code is hard to understand
1
Code smells priority in a project
3
Developers strive to use 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

- Most 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.

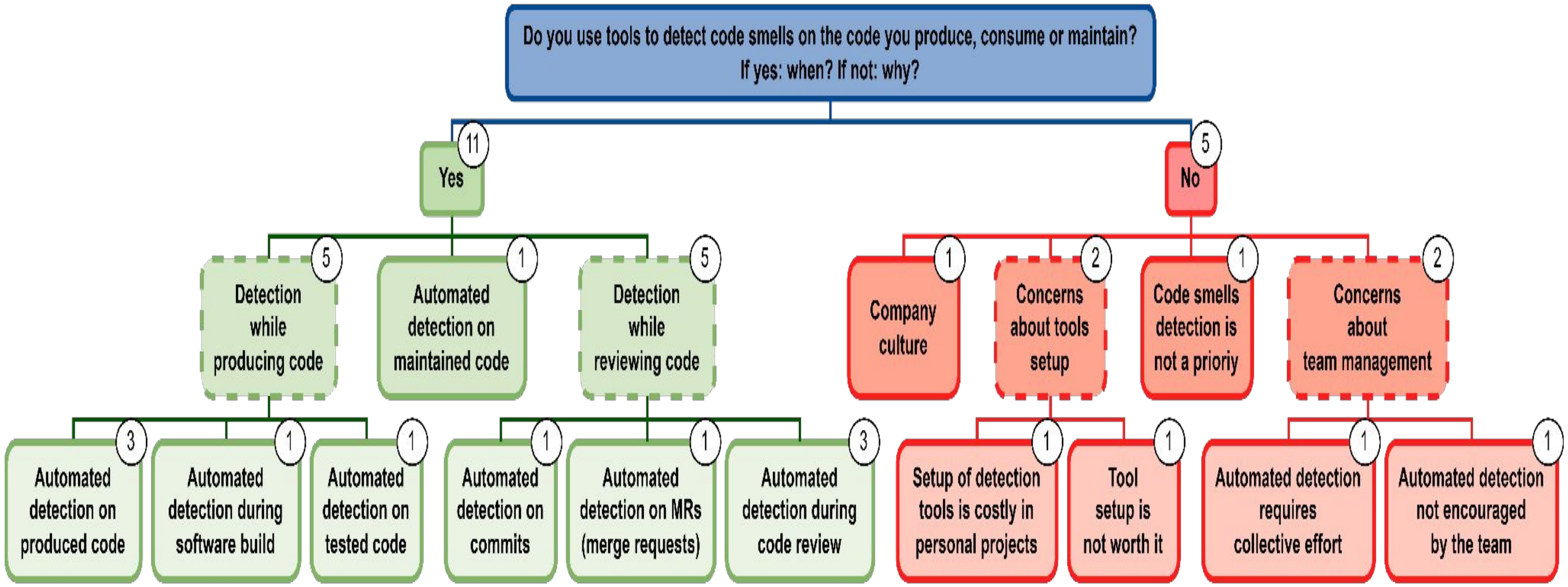
Results - RQ2

- Most 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, some of them claimed that it depends on other factors, like feature prioritization.

Results - RQ2

- Most 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, some of them claimed that it depends on other factors, like feature prioritization.
- They were also mixed on their perception about their teammates’ concerns.

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, 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, WhatsApp and LinkedIn as the tools for the interviews.

Threats to validity

- **Internal Validity**: Using Telegram, WhatsApp and LinkedIn as the tools 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, WhatsApp and LinkedIn as the tools 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, WhatsApp and LinkedIn as the tools 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, WhatsApp and LinkedIn as the tools 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, WhatsApp and LinkedIn as the tools 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 challenging to generalize our study findings.

Threats to validity

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

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

Felipe Ribeiro
Eduardo Fernandes
Eduardo Figueiredo