How Do Developers Select and Prioritize Code Smells? A Preliminary Study

Natthawute Sae-Lim, Shinpei Hayashi, and Motoshi Saeki

Department of Computer Science
School of Computing
Tokyo Institute of Technology

INTRODUCTION

Code smell[1]

An indicator of a design flaw or a problem in the source code

- One of the factors that cause technical debt ☹
- Increases code component's fault-proneness ☹

Data Class

"Classes that have fields, getting and setting methods for the fields, and nothing else."

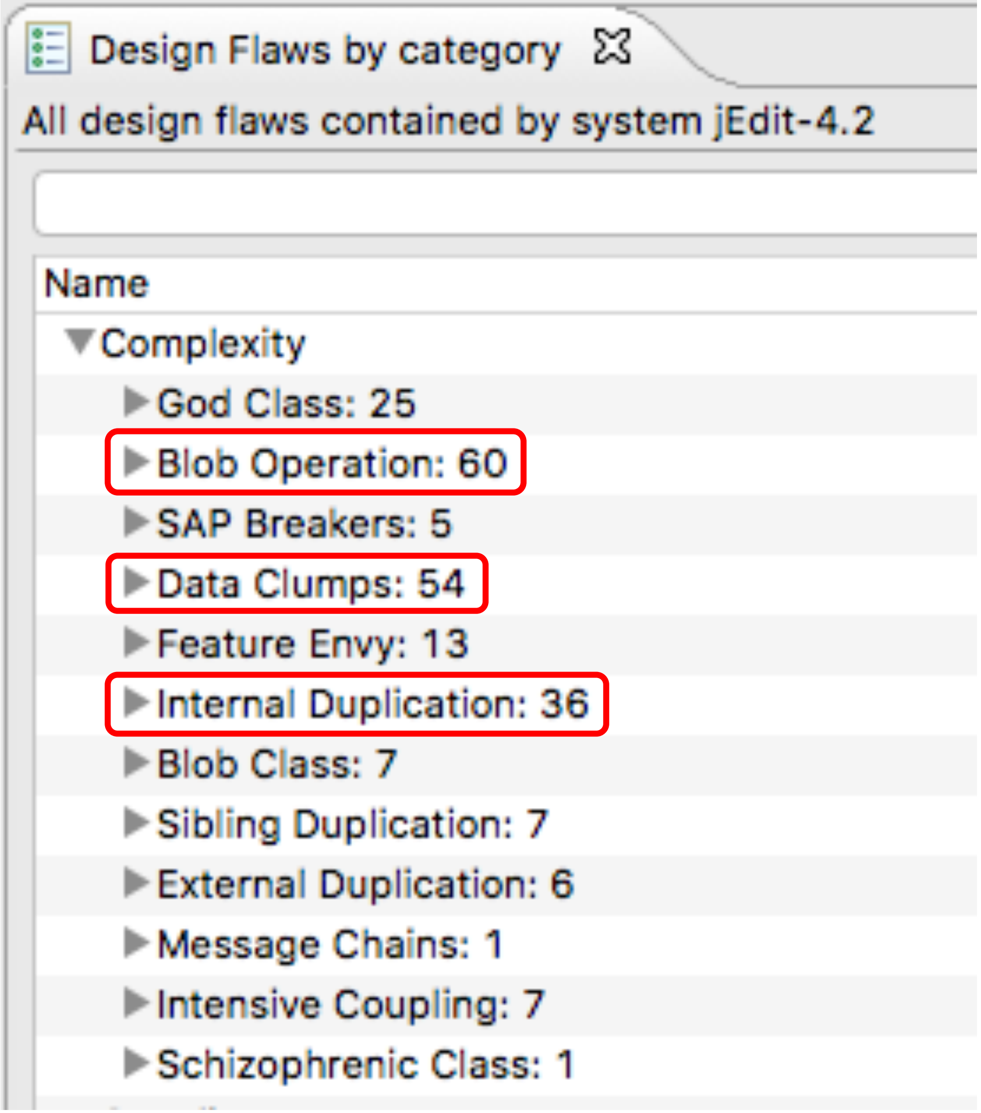
Feature Envy

"Every time you make a kind of change, you have to make a lot of little changes to a lot of different classes."

[1] M. Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

Problem

The number of code smell is **overwhelming**



Design Flaws by category

All design flaws contained by system jEdit-4.2

Name

- ▼ Complexity
  - ▶ God Class: 25
  - ▶ Blob Operation: 60
  - ▶ SAP Breakers: 5
  - ▶ Data Clumps: 54
  - ▶ Feature Envy: 13
  - ▶ Internal Duplication: 36
  - ▶ Blob Class: 7
  - ▶ Sibling Duplication: 7
  - ▶ External Duplication: 6
  - ▶ Message Chains: 1
  - ▶ Intensive Coupling: 7
  - ▶ Schizophrenic Class: 1

Related Work

Code Smells Prioritization

[ICPC 2016]

Context-Based Code Smells Prioritization for Prefactoring

Sae-Lim *et al.*

[MTD 2015]

Towards a Prioritization of Code Debt: A Code Smell Intensity Index

Fontana *et al.*

Code Smells Filtration

[CSMR 2004]

Using history information to improve design flaws detection

Ratiu *et al.*

[ICSE 2015]

Filtering Code Smells Detection Results

Fontana *et al.*

Related Work

Code Smells Prioritization

Task relevance

Smell severity

Code Smells Filtration

Historical information

False positive

Motivation

Code Smells Prioritization

No empirical evidence on how developers handle code smells

information

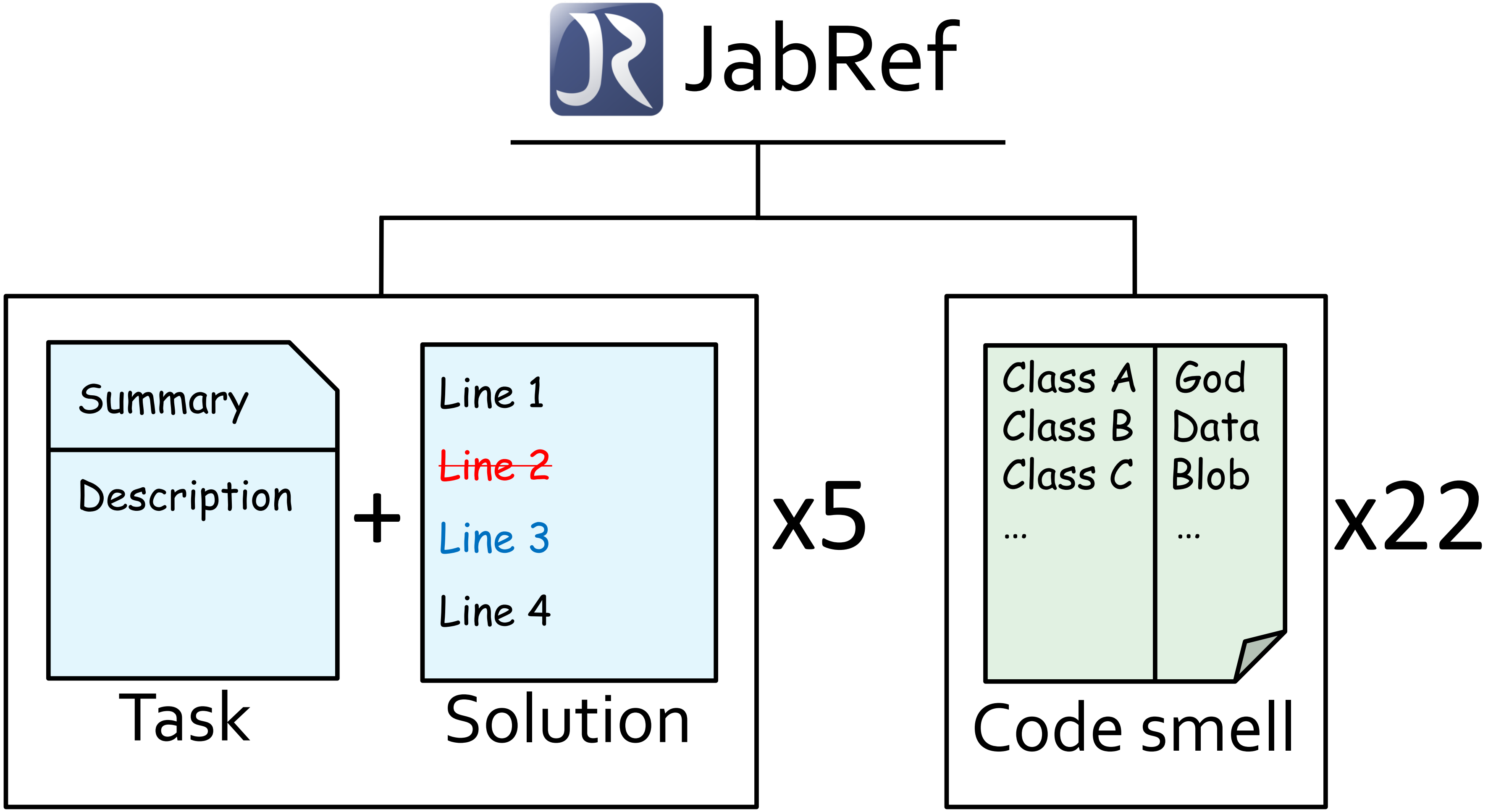
Research Questions

RQ1 : What are the factors used by developers in the code smell selection process?

RQ2 : What are the factors used by developers in the code smell prioritization process?

STUDY DESIGN

Data Collection


JabRef

Summary

Description

+

Line 1

~~Line 2~~

Line 3

Line 4

Task

Solution

x5

Class A	God
Class B	Data
Class C	Blob
…	…

Code smell

x22

Data Collection

Selection

This smell should be solved because …

✔	Class A	God
	Class B	Data
✔	Class C	Blob
	…	…

Code smells

x10

Prioritization

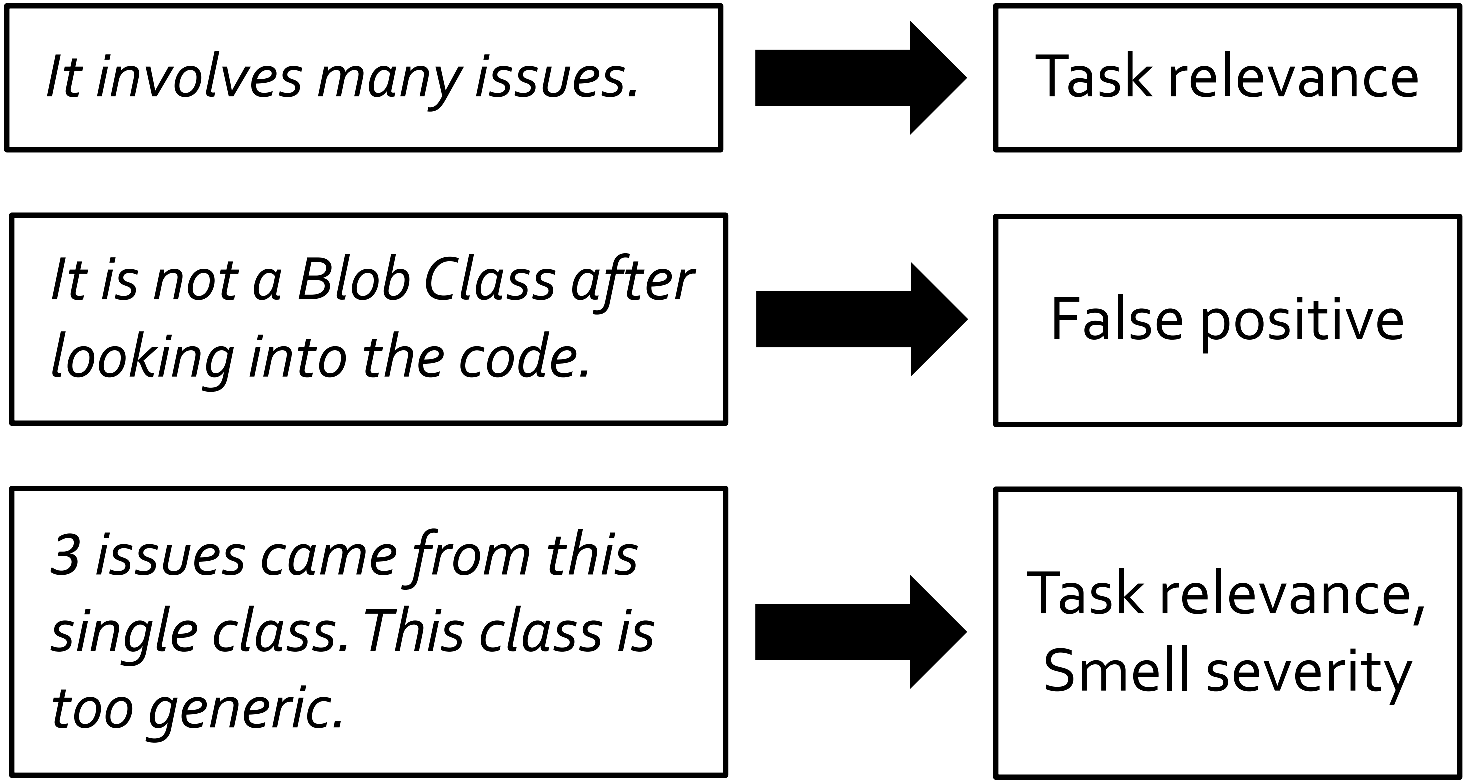
This smell should be solved (in this order) because …

1	Class C	Blob
2	Class A	God
	…	…

Code smells

x10

Coding Technique



Response

Codes

It involves many issues. → Task relevance

It is not a Blob Class after looking into the code. → False positive

3 issues came from this single class. This class is too generic. → Task relevance, Smell severity

RESULTS

15 Final Codes

Smell severity
Smell coupling
Co-located smells
Smell false positive

Task relevance
Task importance
Task implementation cost
Task implementation risk

Testability
Readability
Maintainability
Understandability

Module importance
Module dependency

Refactoring cost

RQ1: Selection Process

Top 5 Factors

Code	Number of responses
Task relevance	33
Smell severity	11
Task implementation cost	5
Testability	5
Co-located smells	4

Factors considered together

Code	Number of responses
Task relevance, Smell severity	9
Task relevance, Testability	5

RQ2: Prioritization Process

Top 5 Factors

Code	Number of responses
Module importance	14
Task relevance	10
Testability	5
Smell severity	4
Maintainability	3

Factors considered together

Code	Number of responses
Module importance, Task relevance	4
Module importance, Testability	3

CONCLUSION

Conclusion

How do developers **select** and **prioritize** code smells?

Selection:	Task relevance
	Smell severity
Prioritization:	Module importance
	Task relevance

Take-home message

Factors that have been considered

Smell severity
Task relevance
Smell false positive

Factors that have not been considered

Testability
Readability
Smell coupling
Maintainability
Task importance
Refactoring cost
Co-located smells
Understandability
Module importance
Module dependency
Task implementation risk
Task implementation cost