Revisiting Context-Based Code Smells Prioritization: On Supporting Referred Context

Natthawute Sae-Lim, Shinpei Hayashi, and Motoshi Saeki



Department of Computer Science
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 ☹

```
public class Employee...
   private void printSalary(){
      int salary;
      if (wHours > 40)
         salary = (40*1000)+((wHours-40)*1500);
      else
         salary = wHours*1000;
      println("Salary: " + salary);
   }
   private vo
      int incomeTax;
      int salary;
      if (wHours > 40)
         salary = (40*1000)+((wHours-40)*1500);
      else
         salary = wHours*1000;
      incomeTax = salary*0.08;
      println("Tax: " + incomeTax);
   }
```

Duplicated Code

Refactoring

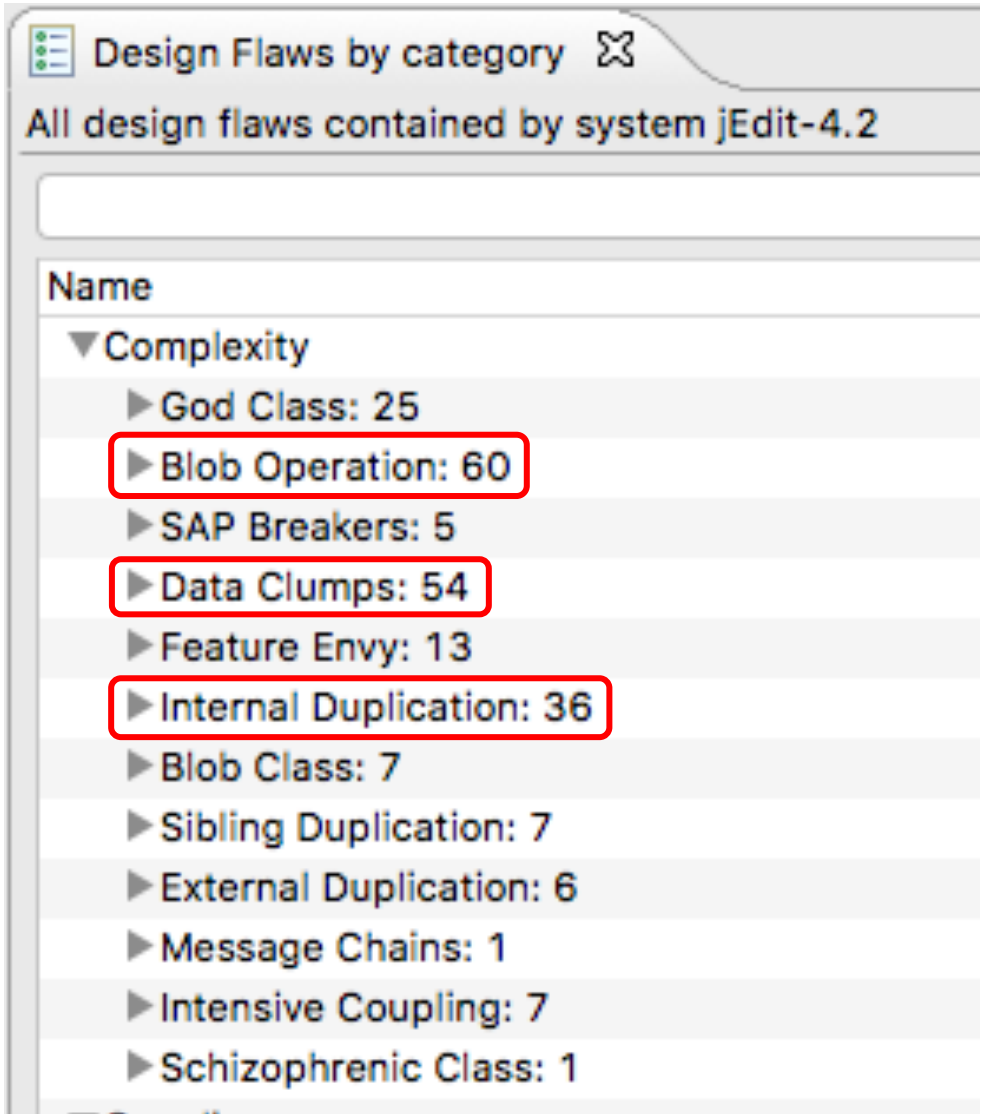
(Extract Method)

```
public class Employee...
   private void printSalary(){
      println("Salary: " + getSalary());
   }
   private void printIncomeTax(){
      int incomeTax = getSalary() * 0.08;
      println("Tax: " + incomeTax);
   }
   private int getSalary(){
      int salary;
      if (wHours > 40)
         salary = (40*1000)+((wHours-40)*1500);
      else
         salary = wHours*1000;
      return salary;
   }
```

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

Problem

The number of code smell is **overwhelming**



Code Smells Prioritization

[J. ASE 2016]

An Approach to Prioritize Code Smells for Refactoring

Vidal *et al.*

[MTD 2015]

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

Fontana *et al.*

[ICPC 2016]

Context-Based Code Smells Prioritization for Prefactoring

Sae-Lim *et al.*

[SBSE 2013]

Prioritization of Code Anomalies based on Architecture Sensitiveness

Arcoverde *et al.*

[ICSE 2016]

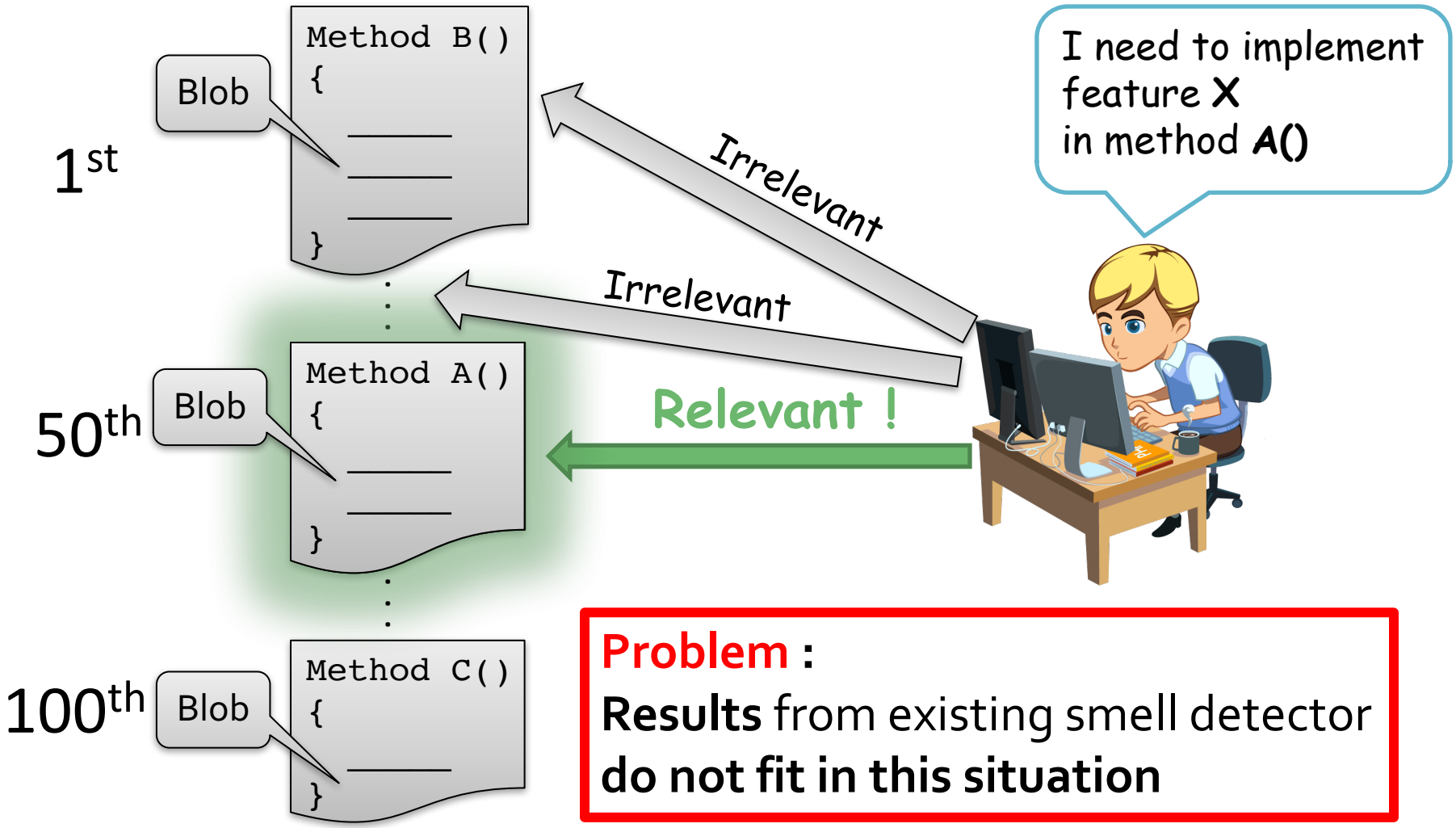
Technical Debt Prioritization using Predictive Analytics

Codabux *et al.*

CONTEXT-BASED CODE SMELLS PRIORITIZATION

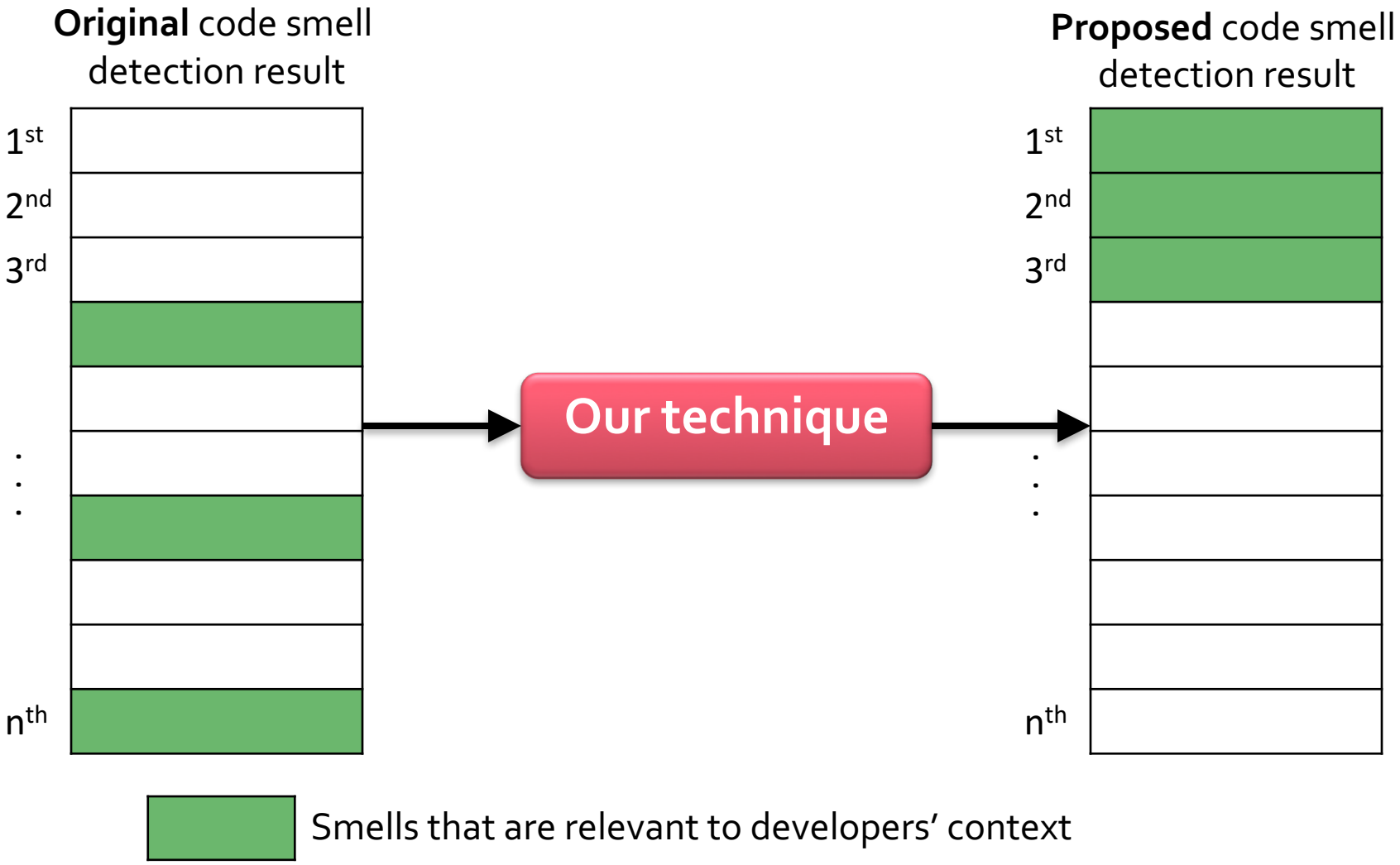
Problem

Code smell detection results

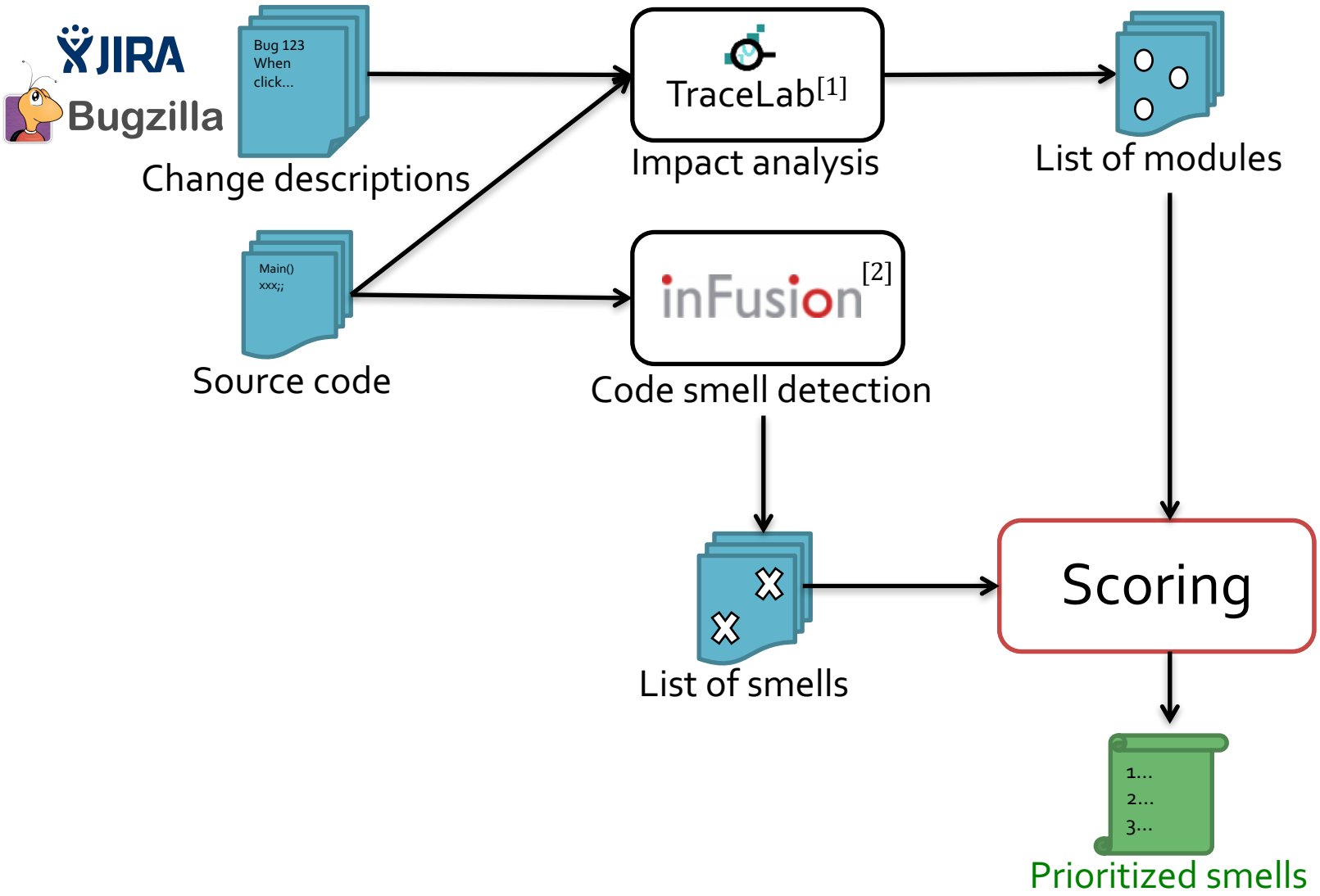


Problem :
Results from existing smell detector **do not fit in this situation**

Goal


Original code smell detection result
1st
2nd
3rd
nth
Our technique
Proposed code smell detection result
1st
2nd
3rd
nth
Smells that are relevant to developers' context

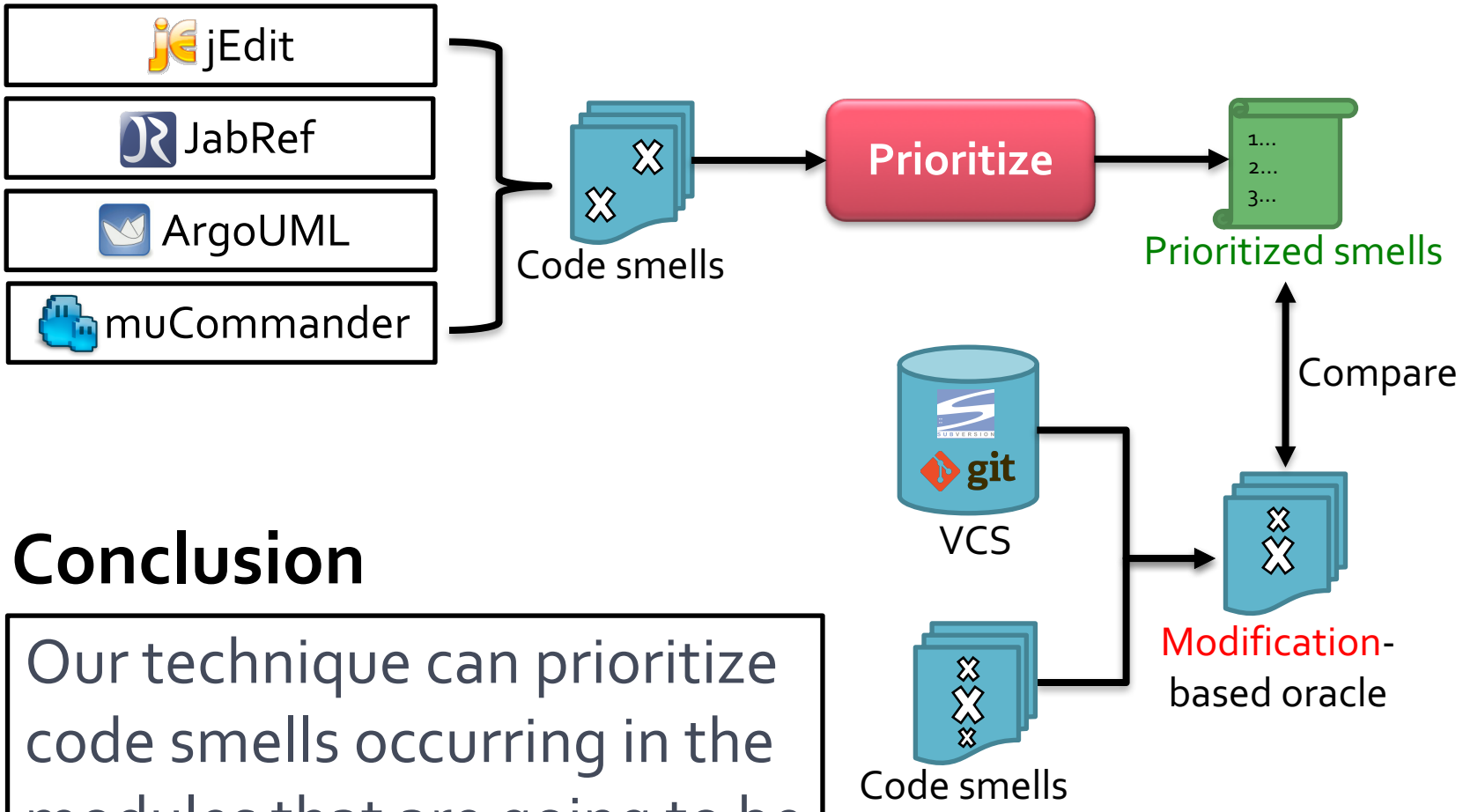
Approach overview



[1] B. Dit, E. Moritz, and D. Poshyvanyk, "A TraceLab-based Solution for Creating, Conducting, and Sharing Feature Location Experiments,", ICPC2012

[2] https://www.intooitus.com/products/infusion

Empirical Study



Conclusion

Our technique can prioritize code smells occurring in the modules that are going to be **modified**

Software change process[1]

Interaction History

Version Control System

Initiation

Concept Location

Identify a module to be modified

Impact Analysis

Identify a *full set* of modules to be modified

Referred Modules

Prefactoring

Actualization

Modify source code

Modified Modules

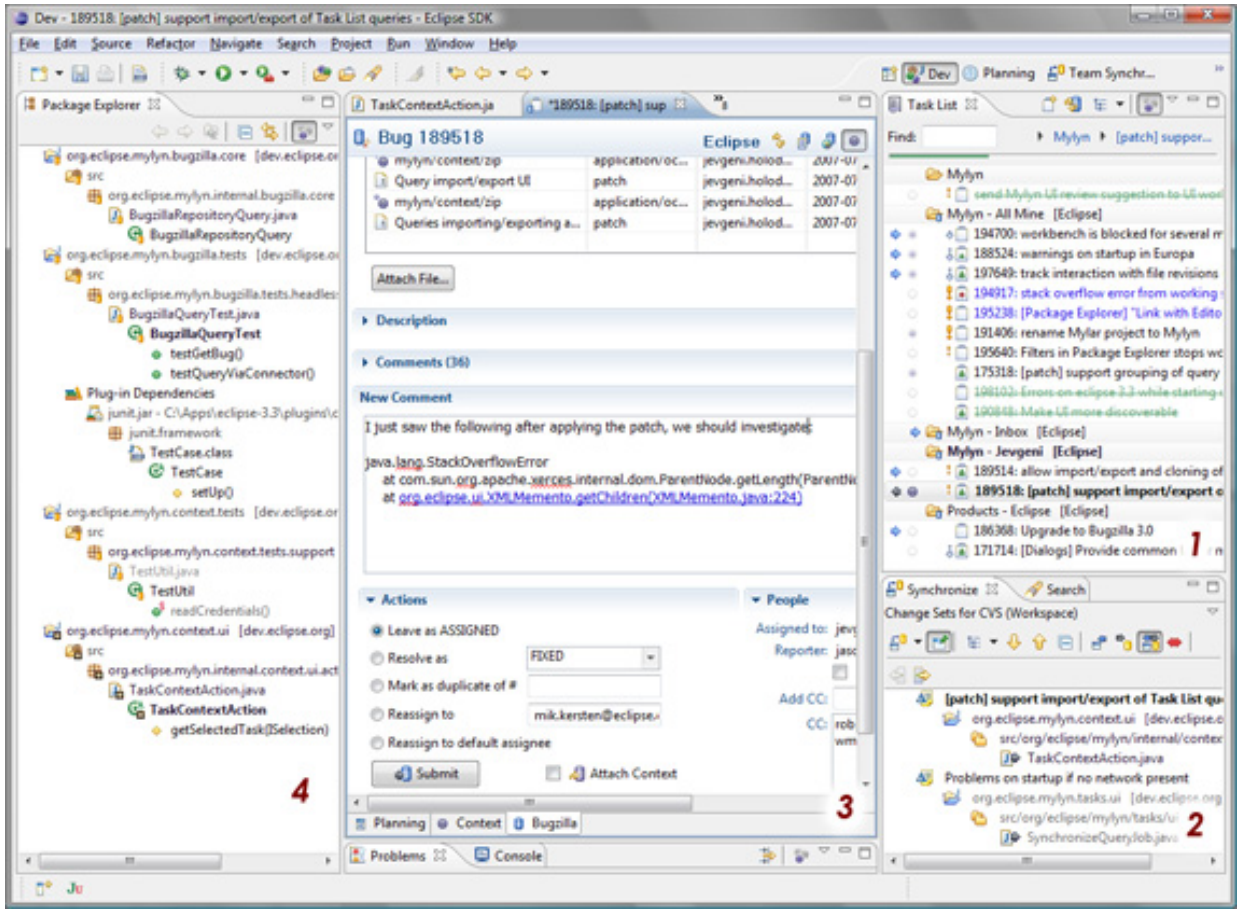
Postfactoring

Conclusion

[1] V. Rajlich, *Software Engineering: The Current Practice*. Chapman and Hall/CRC, 2011

Mylyn

Task and application lifecycle management (ALM) framework for Eclipse.



Mylyn

Developer selects text in editor

```
<InteractionEvent
 Delta="null"
 EndDate="2009-09-08 18:34:51.838 PDT"
 Interest="1.0"
 Kind="edit"
 Navigation="null"
 OriginId="org.eclipse.jdt.ui.CompilationUnitEditor"
 StartDate="2009-09-08 18:34:51.838 PDT"
 StructureHandle="=org.eclipse.mylyn.internal.context.
                  ui{IContextUiHelpIds.java"
 StructureKind="java”
/>
```

Software change process[1]

Interaction History

Version Control System

Initiation

Concept Location

Identify a module to be modified

Impact Analysis

Identify a *full set* of modules to be modified

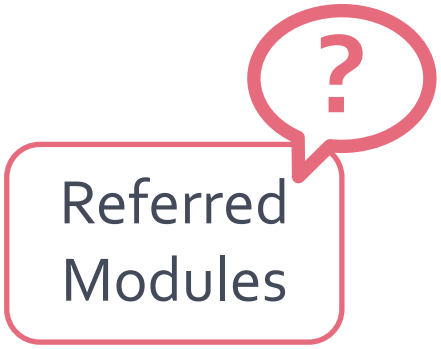
Prefactoring

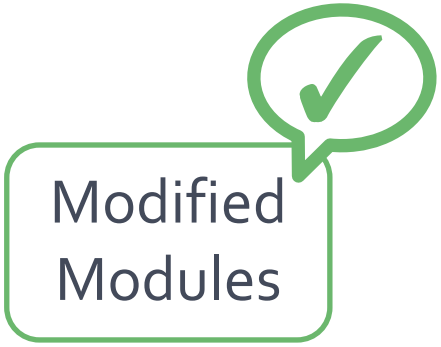
Actualization

Modify source code

Postfactoring

Conclusion





[1] V. Rajlich, *Software Engineering: The Current Practice*. Chapman and Hall/CRC, 2011

Software change process[1]

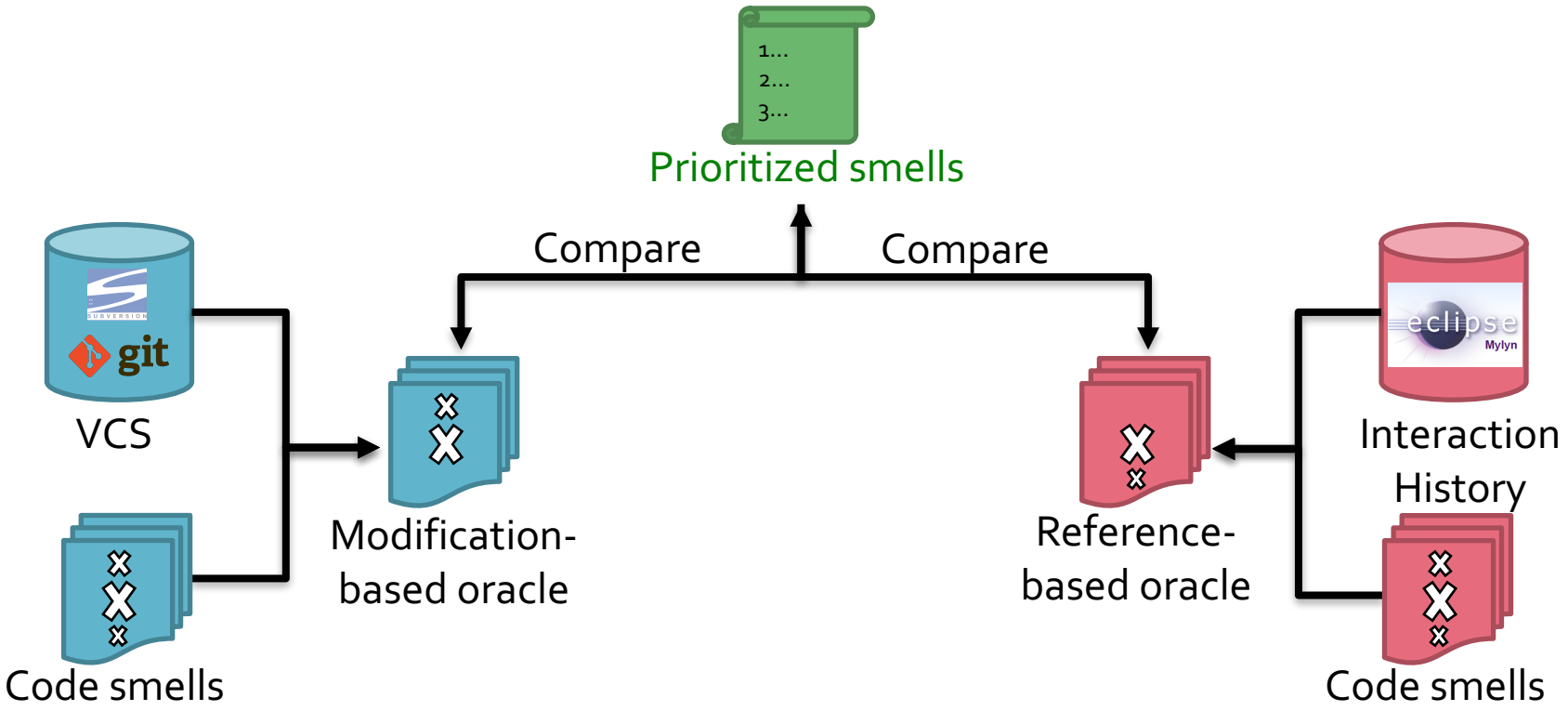
Is our technique useful for referred context?

[1] V. Rajlich, *Software Engineering: The Current Practice*. Chapman and Hall/CRC, 2011

EMPIRICAL STUDY

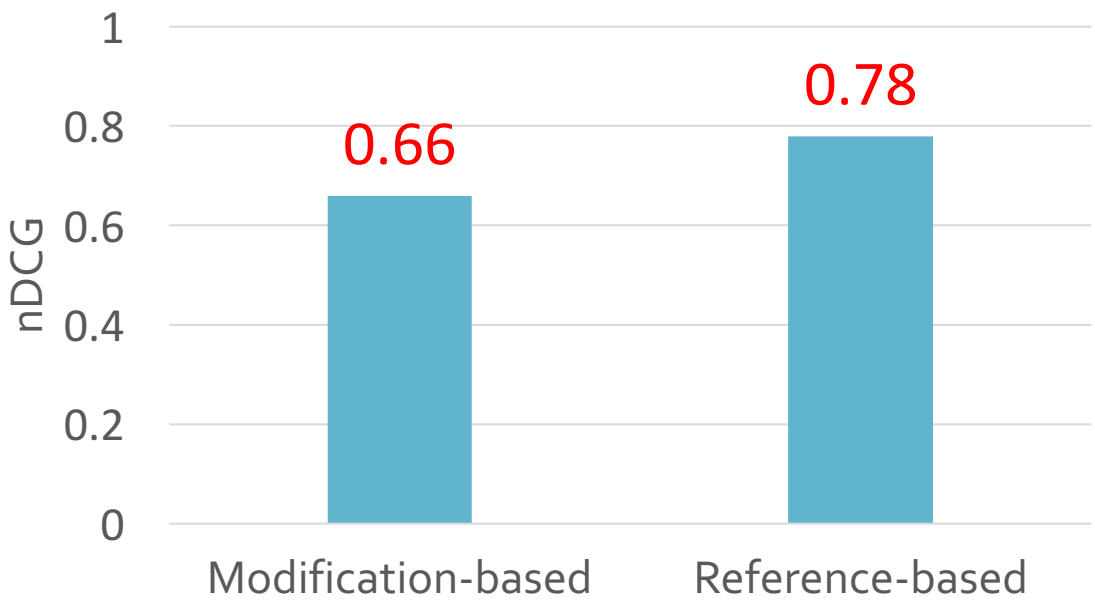
Overview

Subject: Mylyn Task 3.07-3.21



Result

- Is our technique useful for referred modules?



Our technique can be useful to support both ***modified modules*** and ***referred modules***

Top 10 results

Rank	Smell Type	Class Name	#RI	#MI
1	God	TasksUiInternal	7	4
2	God	TasksUiPlugin	9	3
3	God	TaskListIndex	3	1
4	God	AbstractTaskEditorPage	3	2
5	God	TaskDataManager	4	2
6	God	TracRepositoryConnector	1	1
7	God	AttachmentUtil	4	1
8	God	SynchronizeTasksJob	3	0
9	Data	TaskData	3	0
10	God	BugzillaRepositoryConnector	0	2

#RI = Number of referring issues
#MI = Number of modifying issues

CONCLUSION

Messages

Context-based code smells prioritization

Modified Context

Referred Context

Can support both types of context