

# 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<sup>[1]</sup>


*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 void printIncomeTax(){
        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;
    }
}
```



# 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	



# 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**

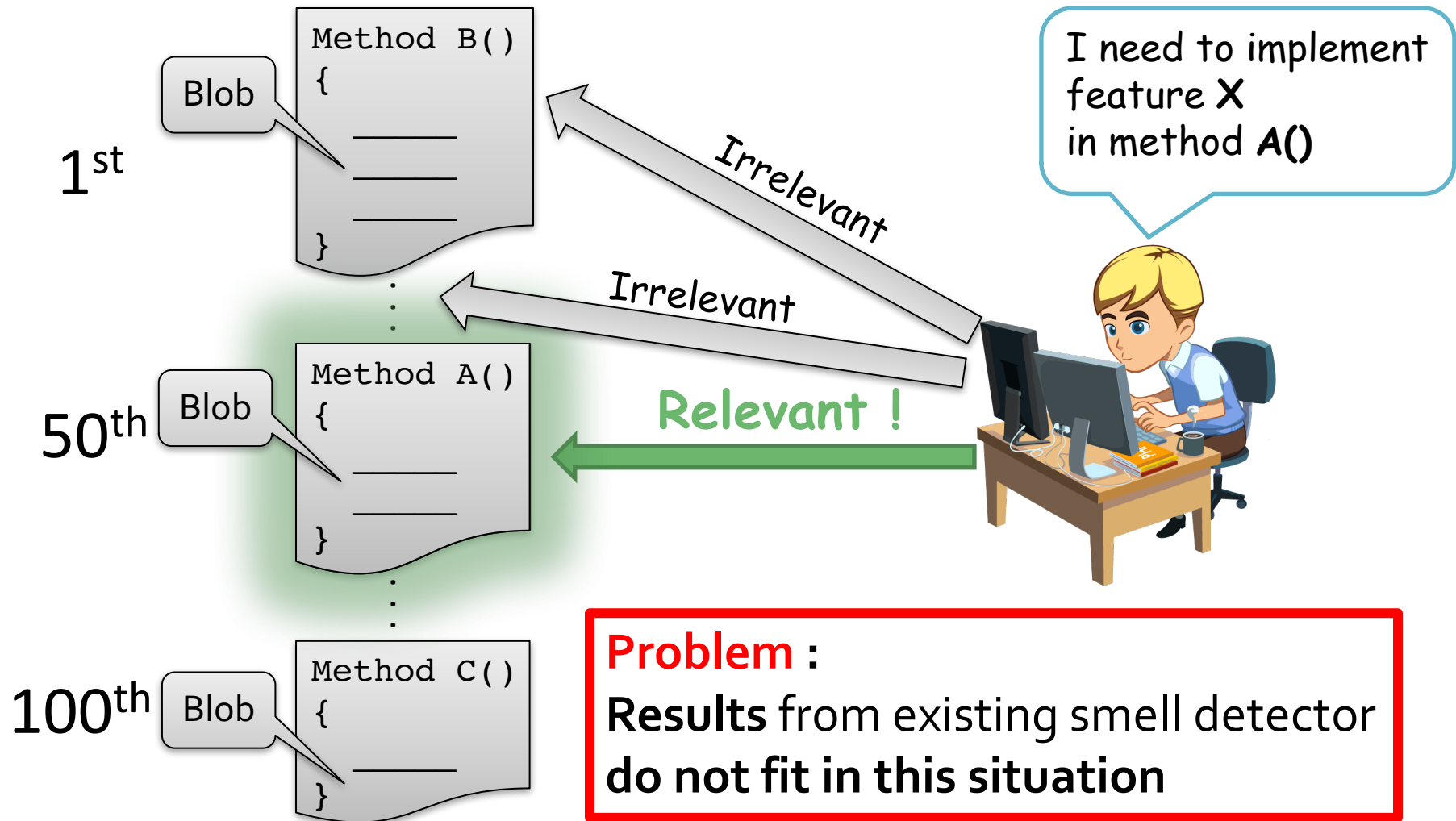
Codabux *et al.*



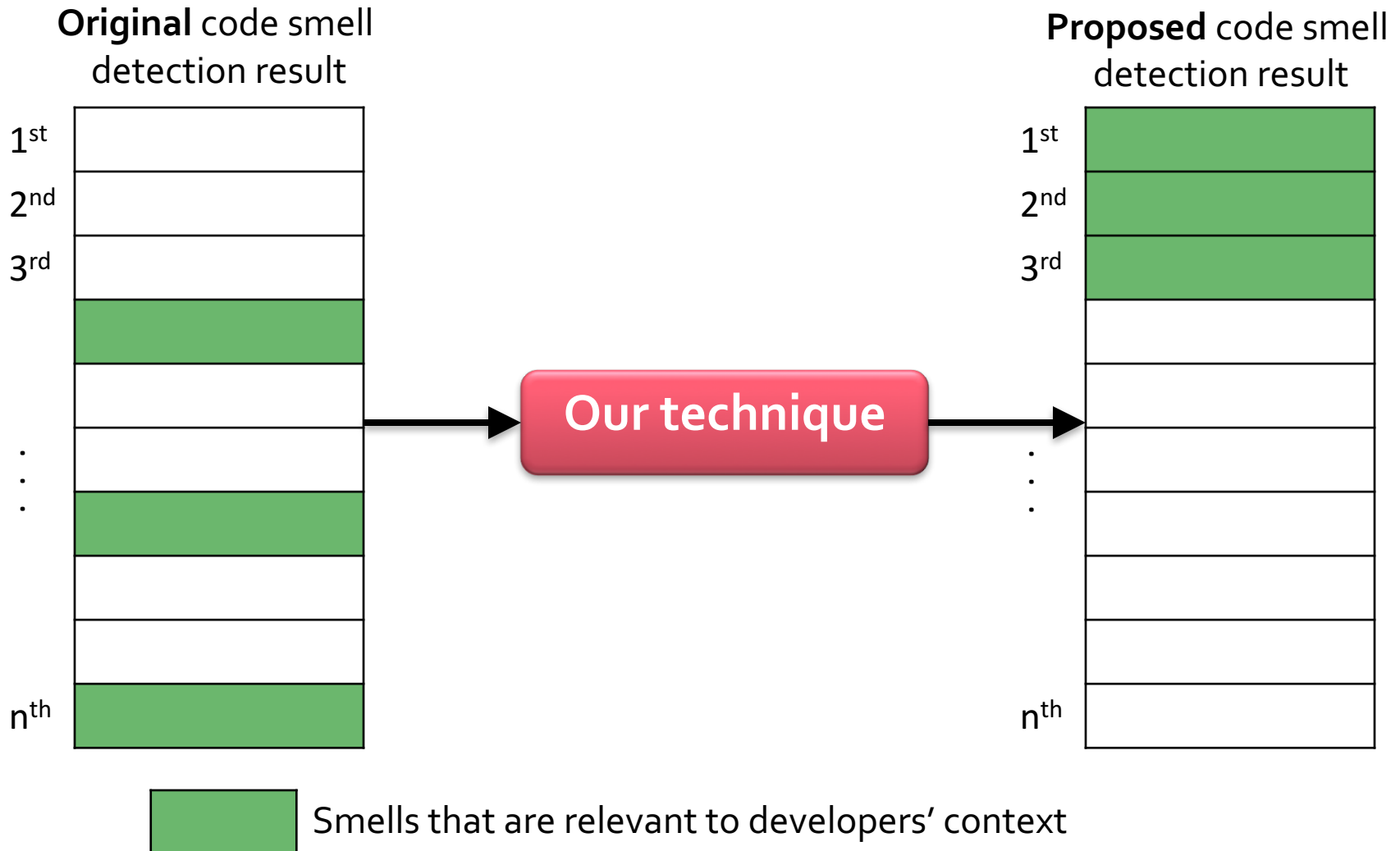
# **CONTEXT-BASED CODE SMELLS PRIORITIZATION**

# Problem

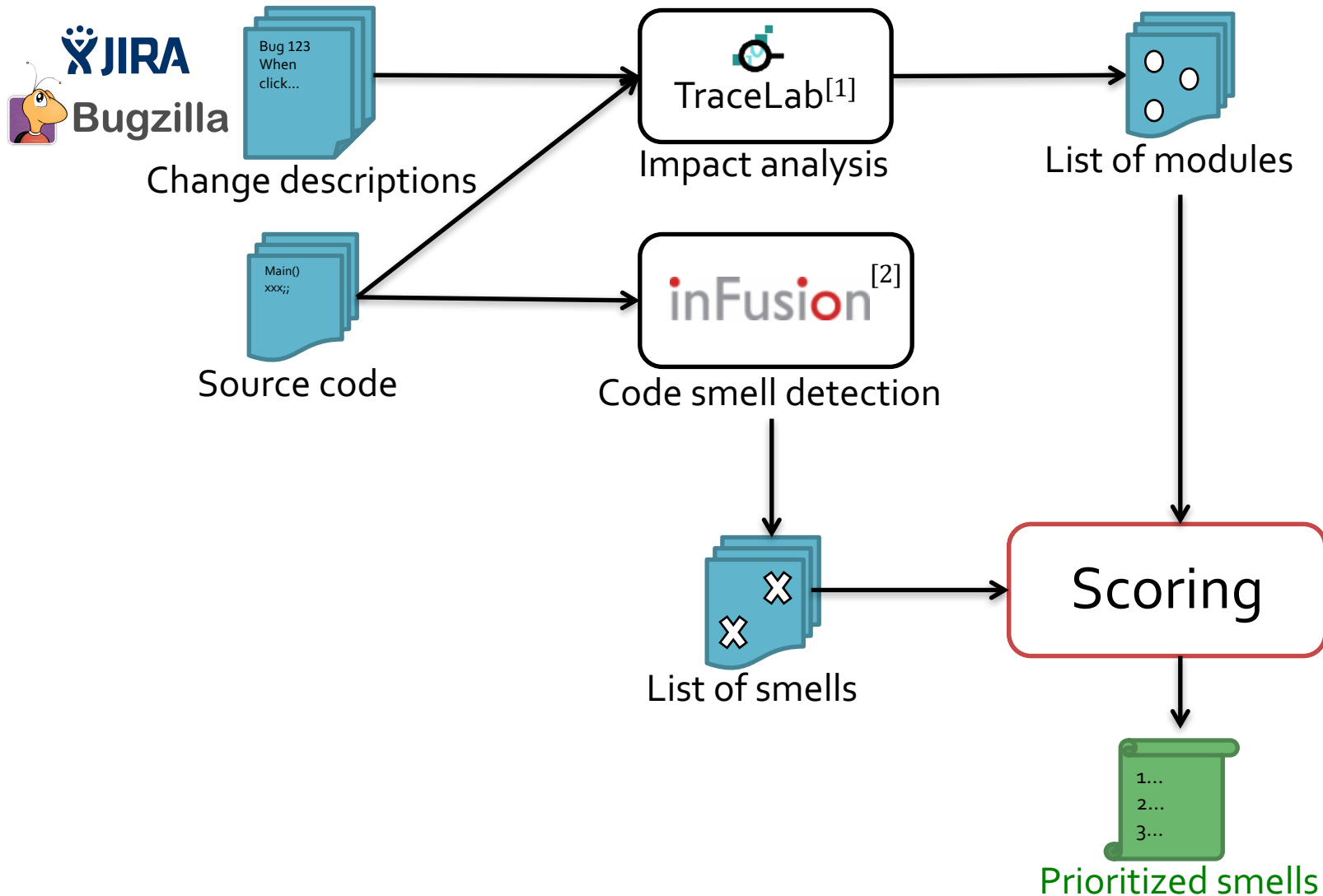
## Code smell detection results



# Goal



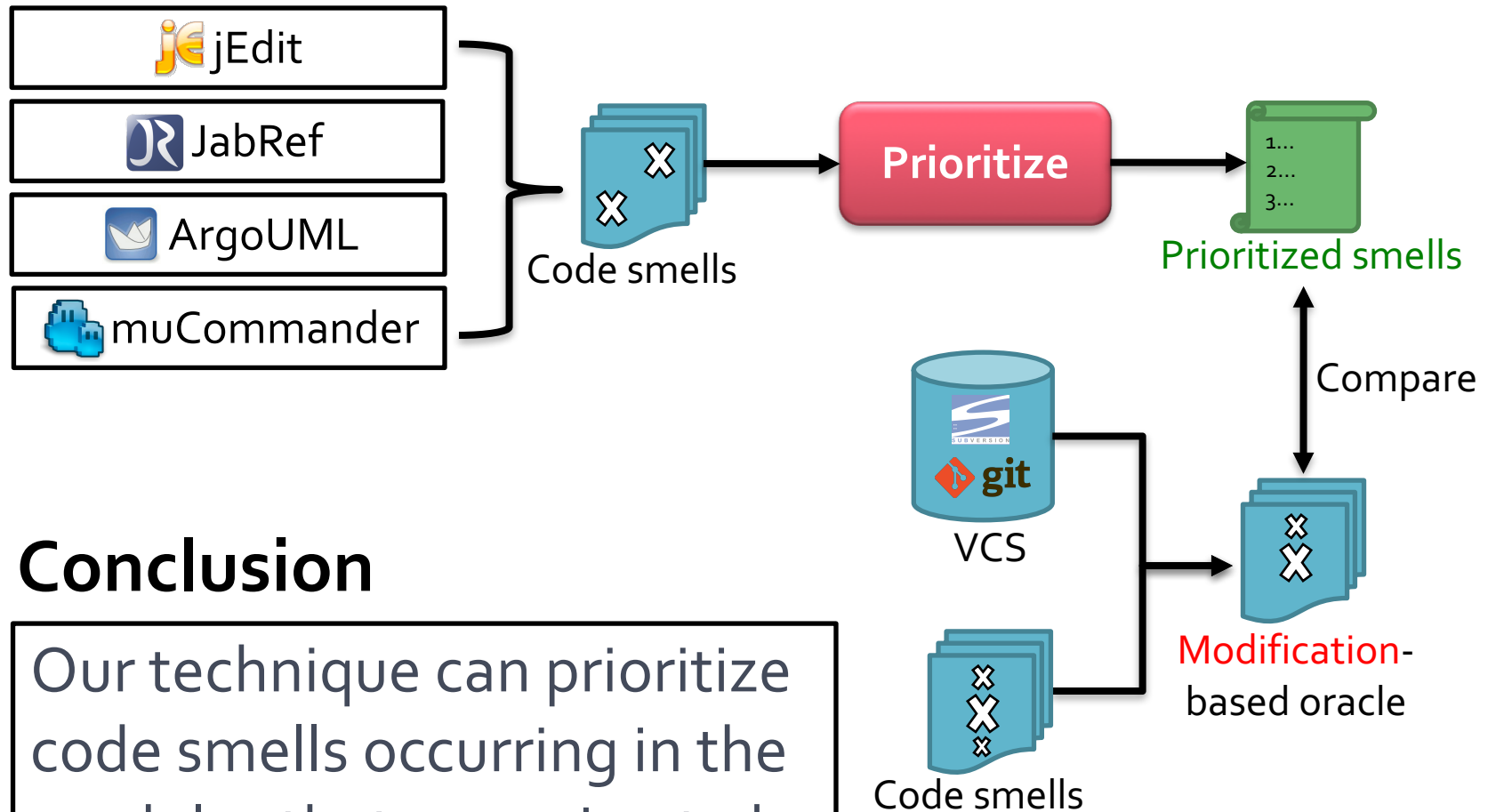
# Approach overview



[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>

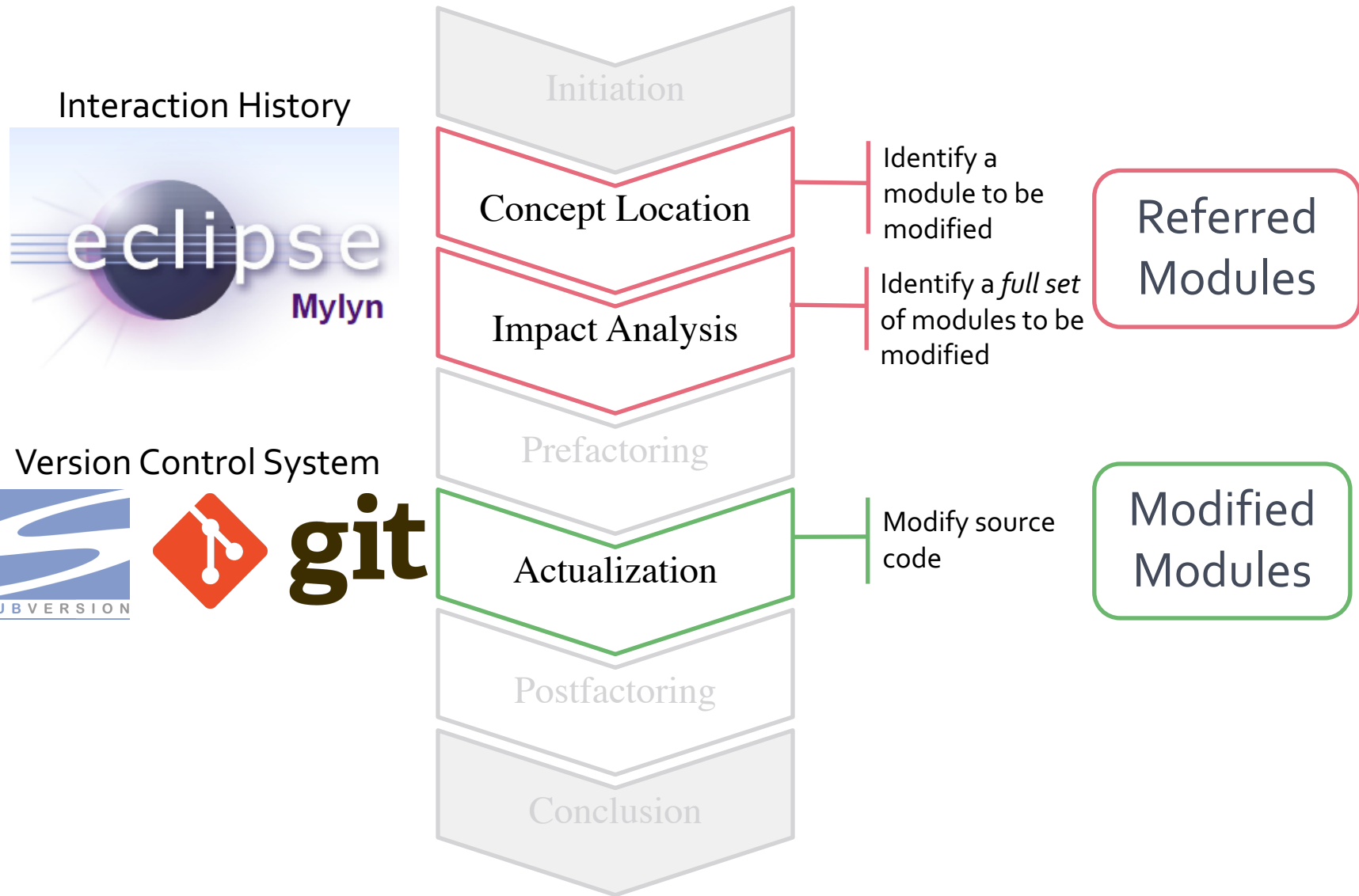
# Empirical Study



## Conclusion

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

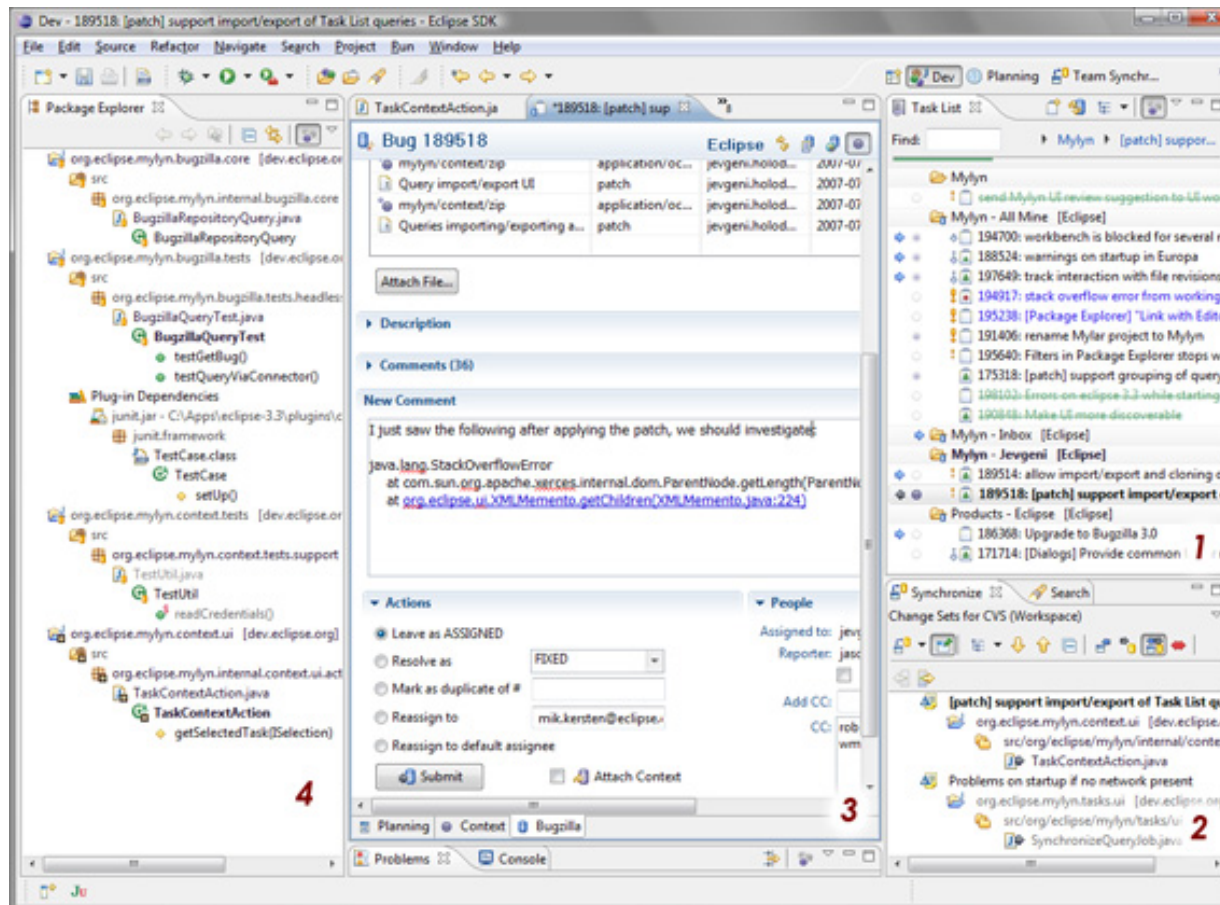
# Software change process<sup>[1]</sup>





# Mylyn

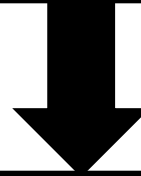
Task and application lifecycle management (ALM) framework for Eclipse.



# Mylyn

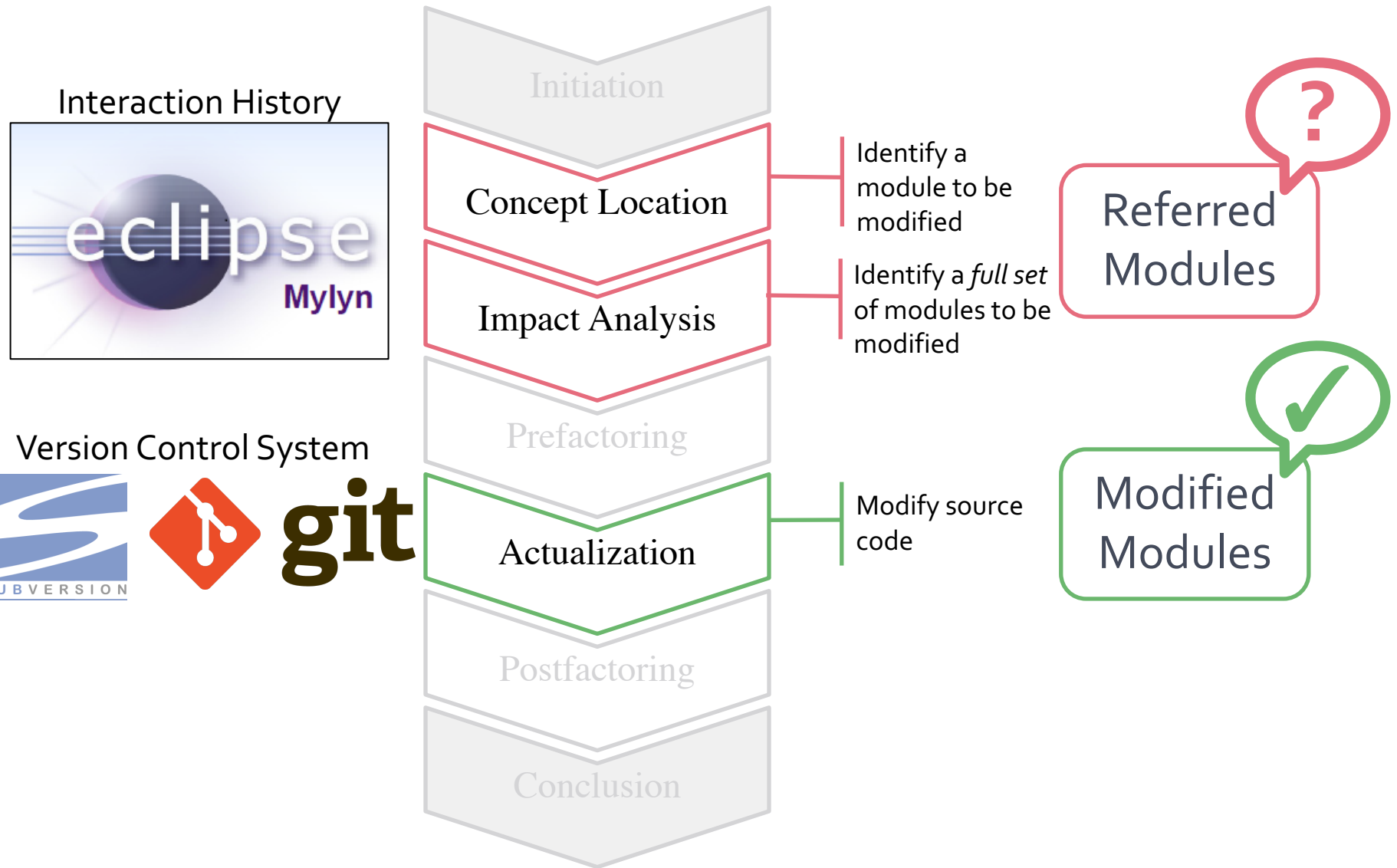


Developer selects text in editor



```
<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<sup>[1]</sup>



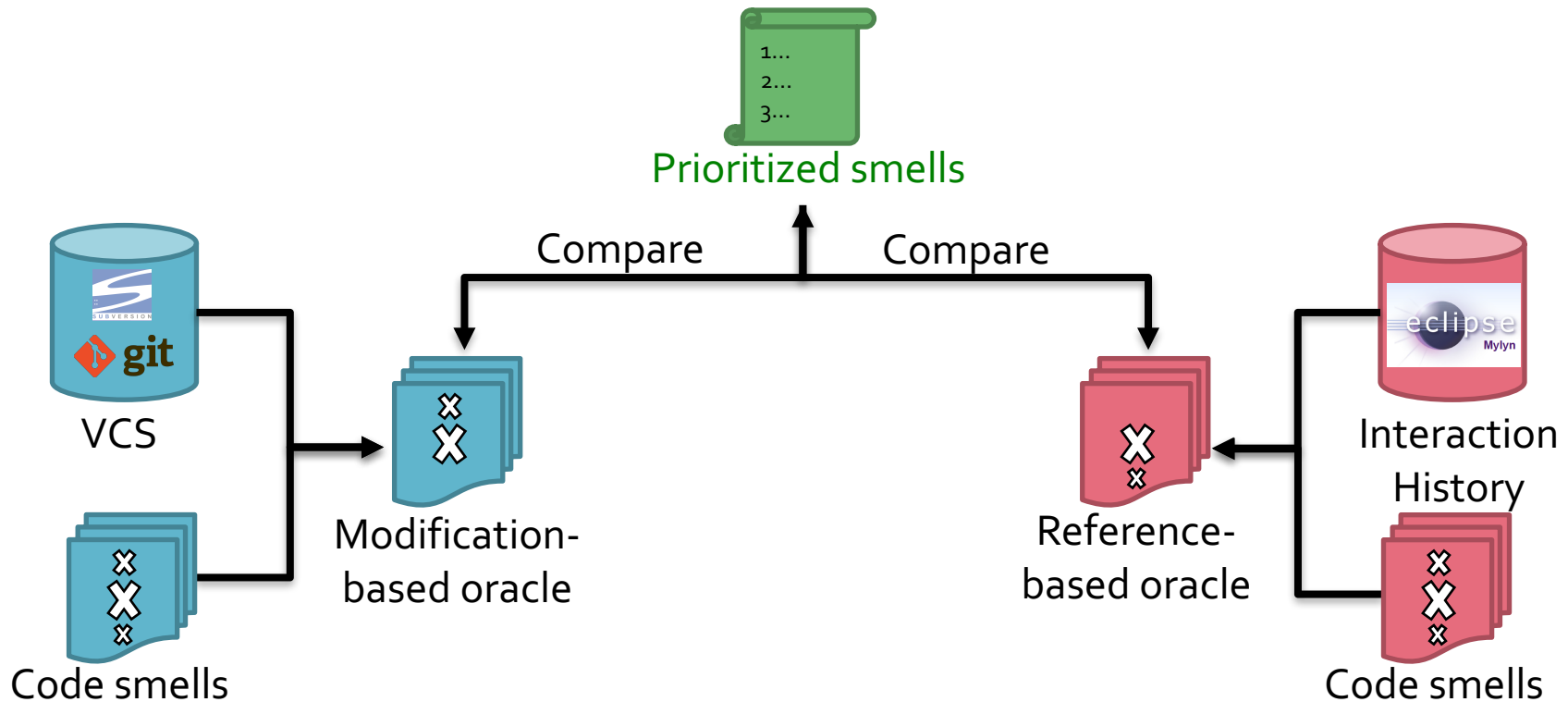
**Is our technique  
useful for  
referred context?**



# **EMPIRICAL STUDY**

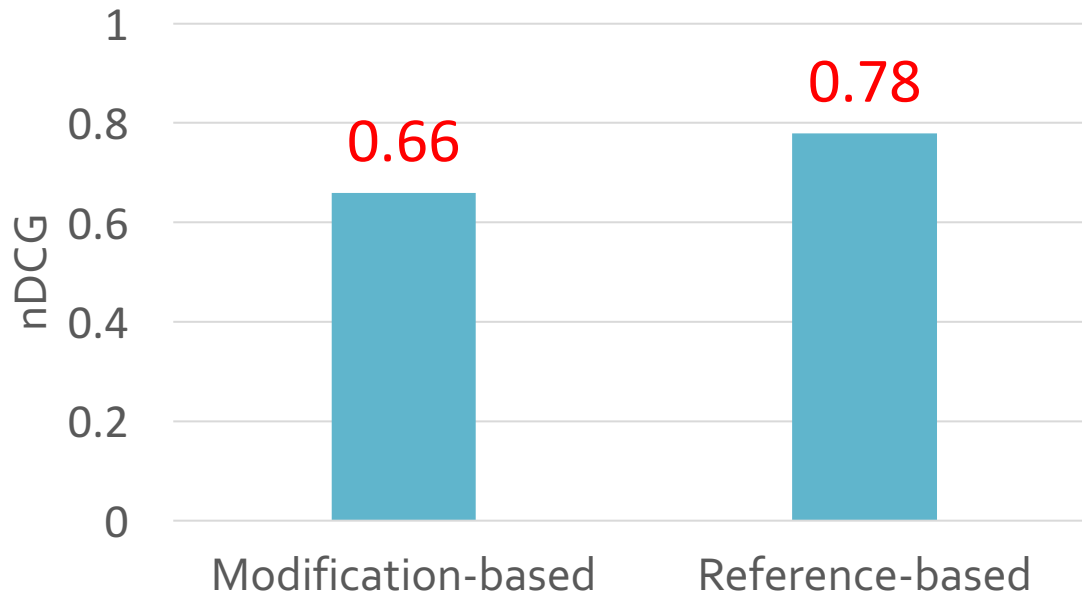
# Overview

**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