Toward Prioritizing Code Smell Detection Results for Prefactoring

Natthawute Sae-Lim, Shinpei Hayashi, Motoshi Saeki

Department of Computer Science
Graduate School of Information Science and Engineering
Tokyo Institute of Technology

INTRODUCTION

Code smell detector

- **Code smell detector**
  - A tool that detect code smells by analyzing source code
  - Suggests refactoring opportunities to developers
- **Problem: Ignoring current context of developer**
  - E.g. "I'm going to implement the XXX feature"
  - The results are mixed with
    - Smells relevant to the context
    - Smells irrelevant to the context
  - Results do not fit **prefactoring** phase

Refactoring **before** implementing feature

Motivating example

Very large and complex

162 LOC

Original result when applying ArgoUML v4.2 to code smell detector

Rank	Smell	Module	Severity
…	…	…	…
8	Intensive Coupling	UmlFactoryMDRImpl.buildNode(Object)	8
…	…	…	…
158	Blob Operation	ProjectBrowser.loadProject(File,boolean)	4
…	…	…	…

Issue #3921

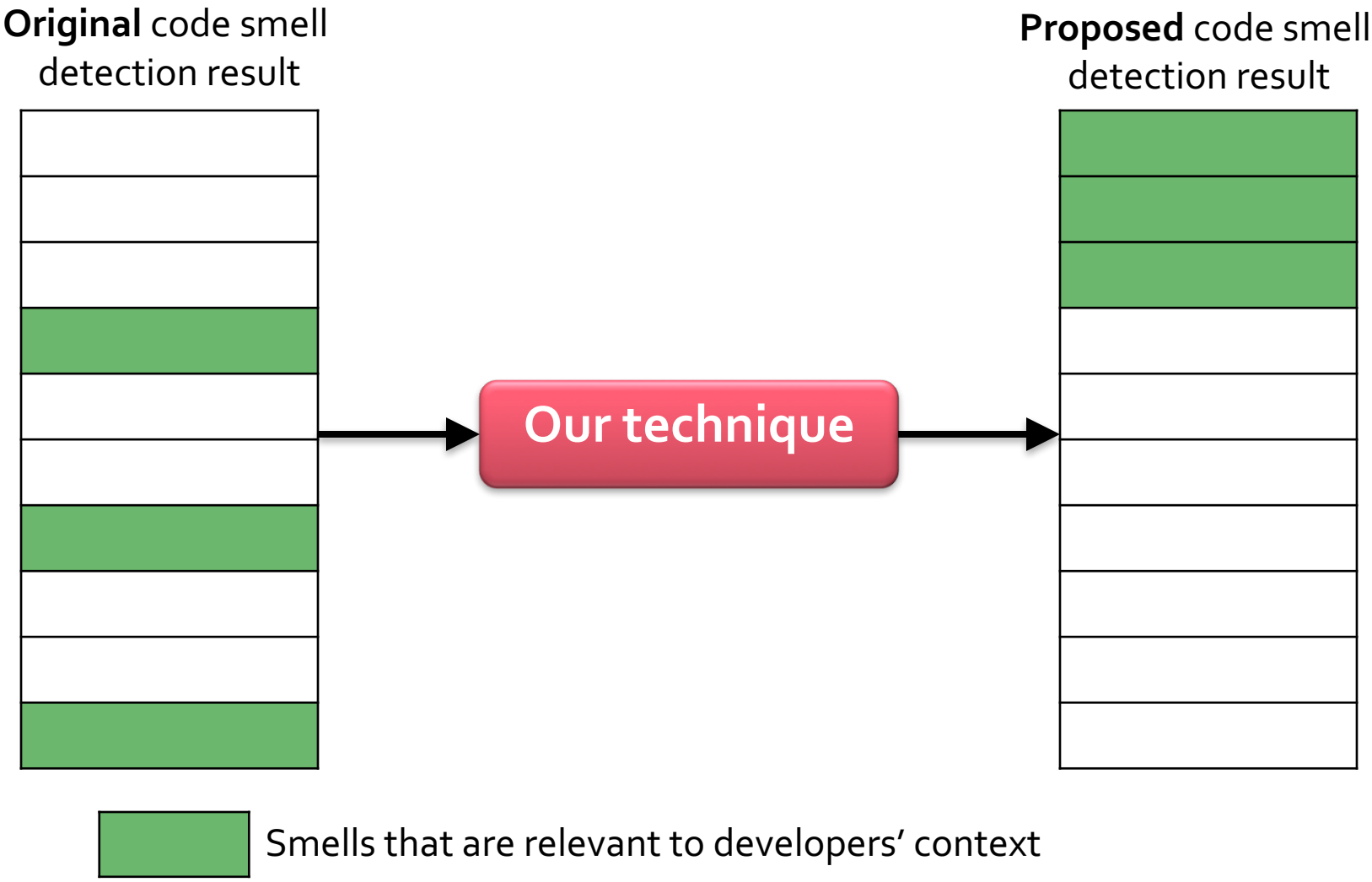
Reopen last saved project should be reopen last project

Issue #4019

Save project dialog should remember what was loaded

Issues to be solved until RELEASE-v2.0

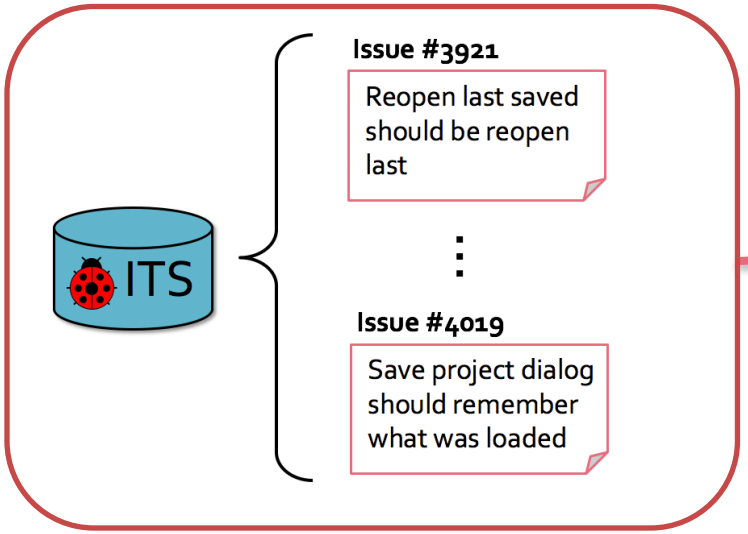
Goal


Original code smell detection result
Our technique
Proposed code smell detection result
Smells that are relevant to developers' context

PROPOSED TECHNIQUE

Developer's context

- **Developer's context = modules to be modified**
- **Issue-driven software development project**
  - Adopting issue tracking system
  - Having list of issues needed to be solved before release



  - This list is used to estimate developer's context

Feature location technique

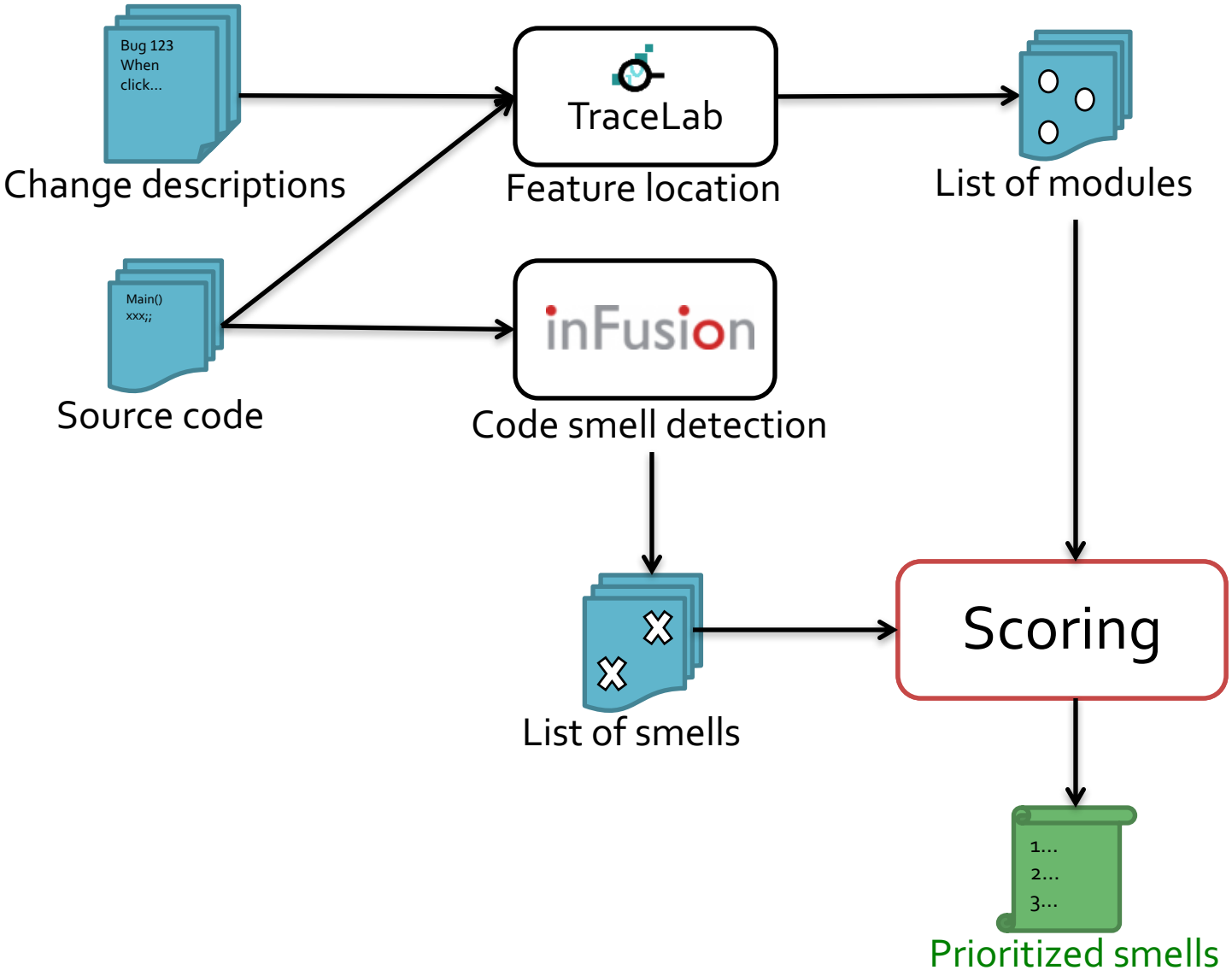
- Identify modules in source code that related to a feature
- Feature location → Change prediction

Change description #3921

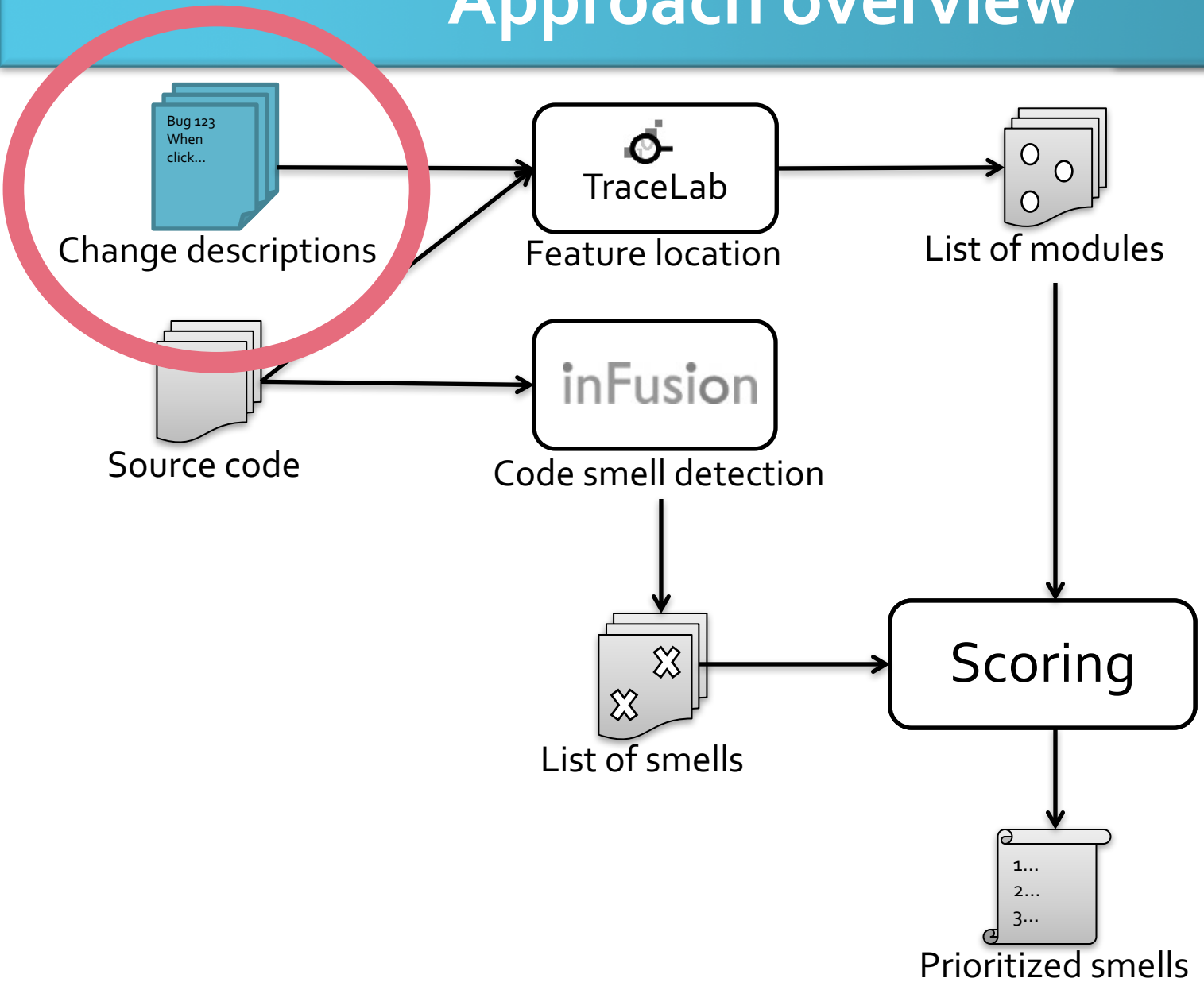
The option to automatically load the last saved project on startup of ArgoUML I think would be more useful as reopen last project (ie last opened project or last saved project)

Relevant modules	Prob
`ArgoParser.getProject()`	0.27
`FigMessage.FigMessage()`	0.26
`ProjectBrowser.loadProject(File,boolean)`	0.22
`ActionSettings.handleSave()`	0.10
…	…
`Project.getBaseName()`	0.04

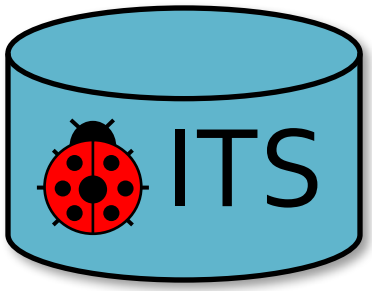
Approach overview



Approach overview



Preparing change descriptions



Issue #3921

Reopen last saved should be reopen last

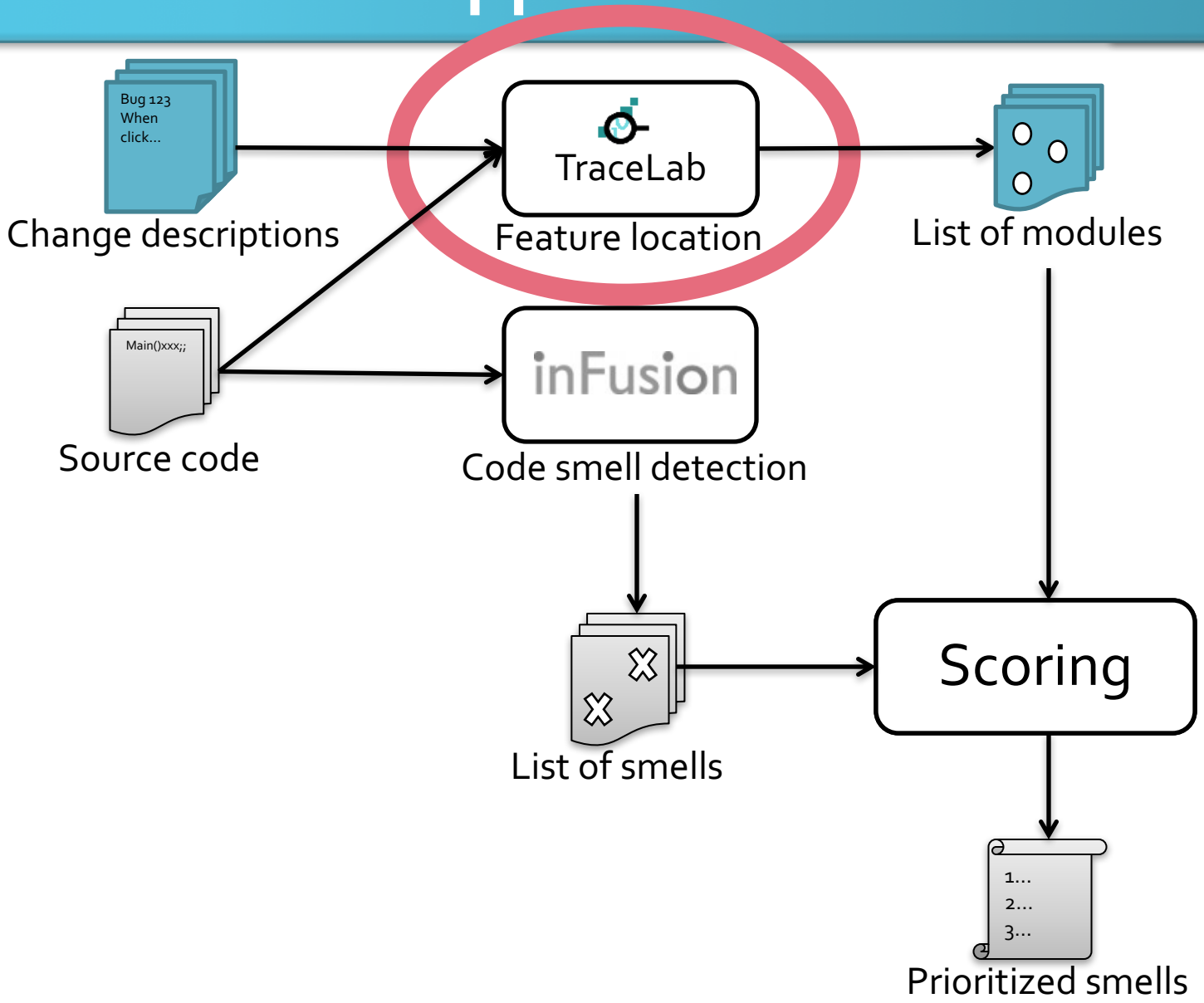
⋮

Issue #4019

Save project dialog should remember what was loaded

Issues to be solved until RELEASE-v2.0

Approach overview



Applying feature location technique

Obtaining relevant sets of modules

Change description #3921

The option to automatically load the last saved project on startup of ArgoUML I think would be more useful as reopen last project (ie last opened project or last saved project)

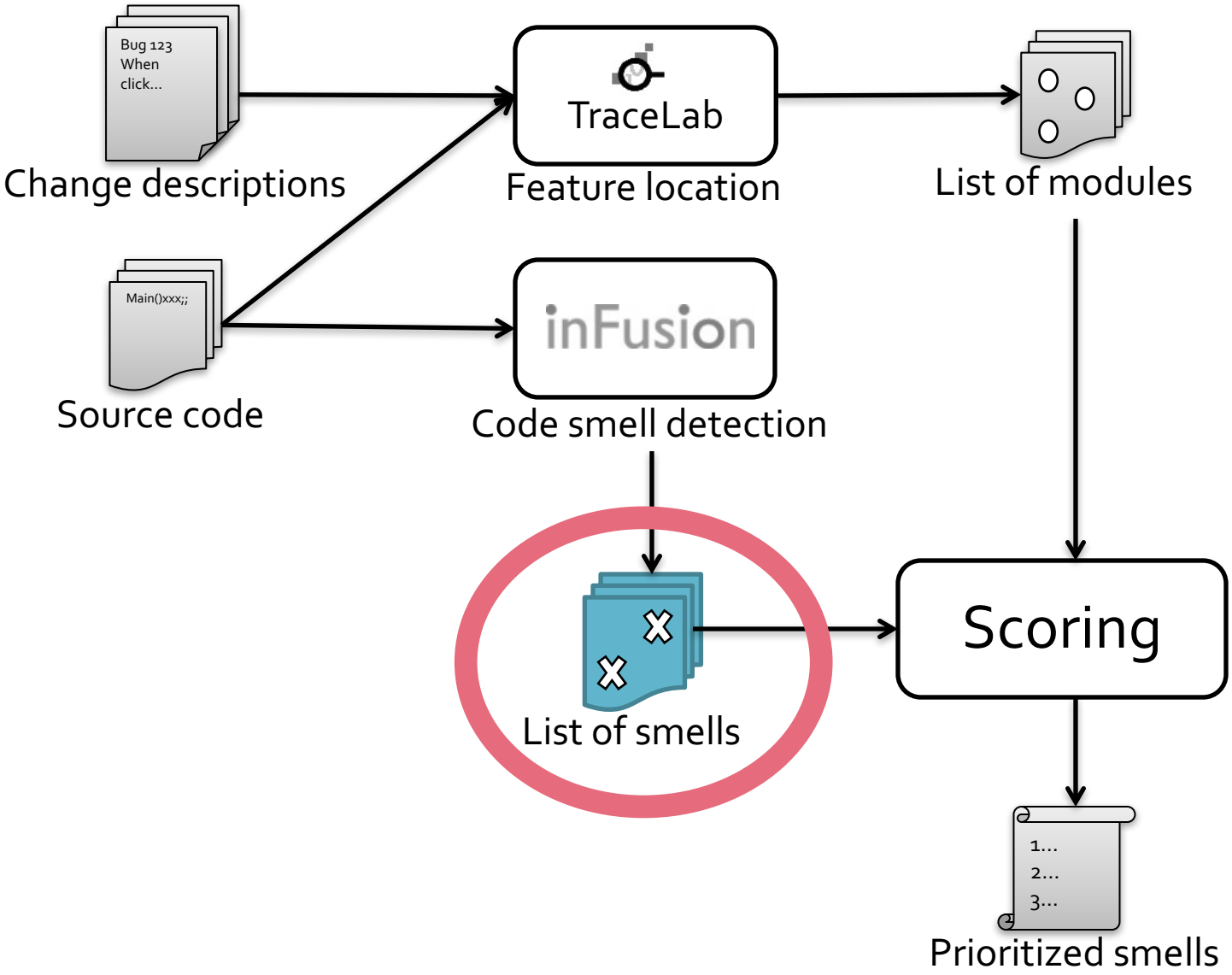
Relevant modules	Prob
`ArgoParser.getProject()`	0.27
`ProjectBrowser.loadProject(File,boolean)`	0.23
`ActionSettings.handleSave()`	0.10
…	…
`Project.getBaseName()`	0.04

Change description #4019

The save project dialog assumes you want to save as the filename of the last project you saved rather than the last project you loaded.

Relevant modules	Prob
`ConfigurationHandler.saveDefault()`	0.36
`ProjectBrowser.loadProject(File,boolean)`	0.22
`CheckMain.getTestModel(String)`	0.15
…	…
`UMLToDoItem.select()`	0.03

Approach overview

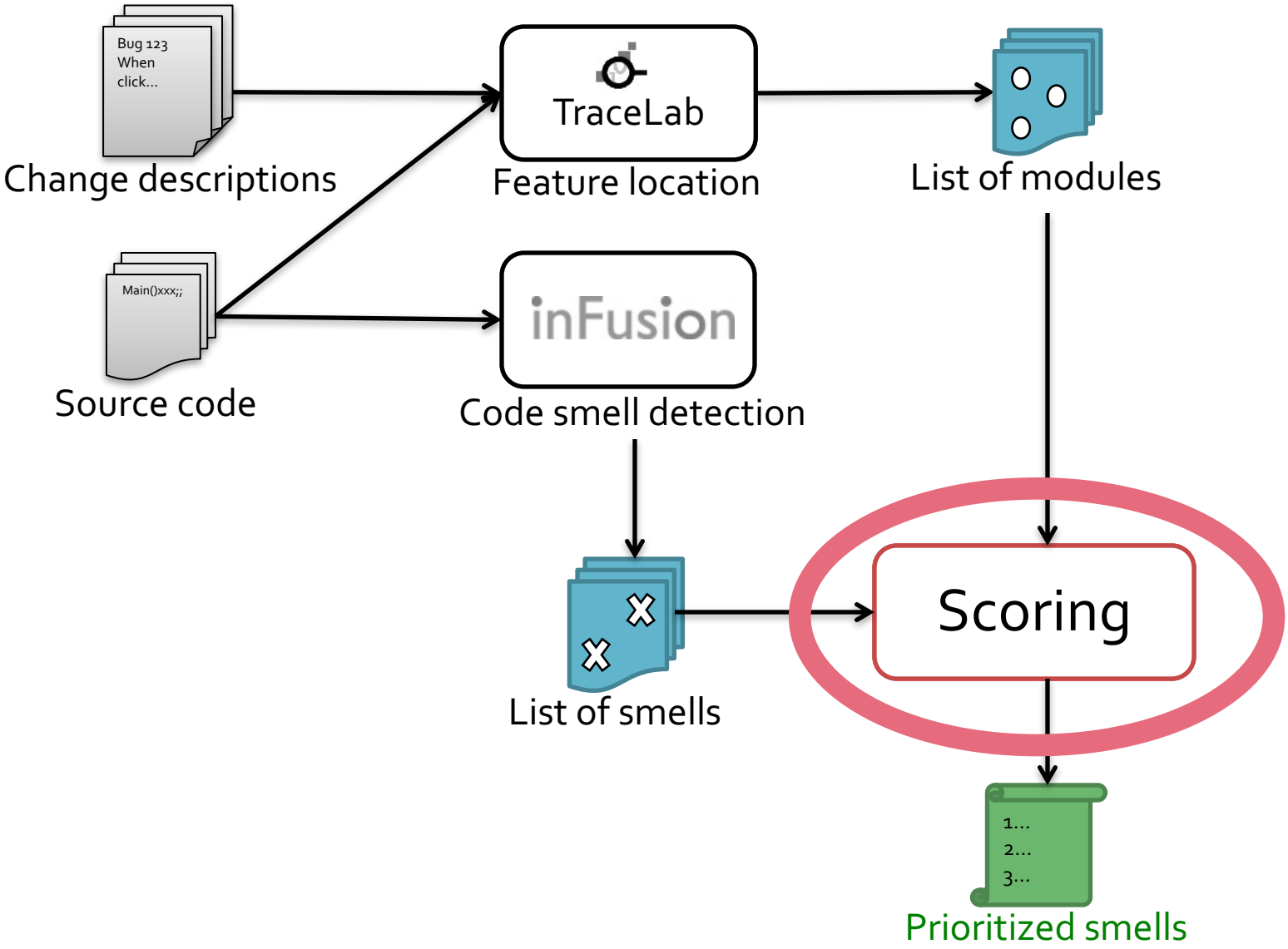


Code smell detection result

Code smell detection result

Smell	Level	Module	Severity
…	Class	…	…
Blob Operation	Method	`ProjectBrowser.loadProject(File,boolean)`	4
…	Subsystem	…	…

Approach overview



Scoring

- Counting matched module in FL result

Code smell detection result

Smell	Level	Module	Score
…	Class	…	…
Blob Operation	Method	`ProjectBrowser.loadProject(File,boolean)`	2
…	Subsystem	…	…

Feature location result

#3921

Relevant modules	Prob
`ArgoParser.getProject()`	0.27
`ProjectBrowser.loadProject(File,boolean)`	0.23
…	…
`Project.getBaseName()`	0.04

#4019

Relevant modules	Prob
`ConfigurationHandler.saveDefault()`	0.36
`ProjectBrowser.loadProject(File,boolean)`	0.22
…	…
`UMLToDoItem.select()`	0.03

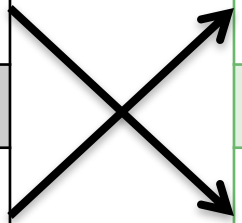
Calculating score

1) Treat every module equally: **Score = 2**
2) Use probability as weight: **Score = 0.45**

Ordering by score

Original code smell detection result

Rank	Smell	Module	Score
…	…	…	…
8	Intensive Coupling	buildNode()	0
…	…	…	…
158	Blob Operation	loadProject()	78
…	…	…	…



Prioritized code smell detection result

Rank	Smell	Module	Score
…	…	…	…
5	Blob Operation	loadProject()	78
…	…	…	…
231	Intensive Coupling	buildNode()	0
…	…	…	…

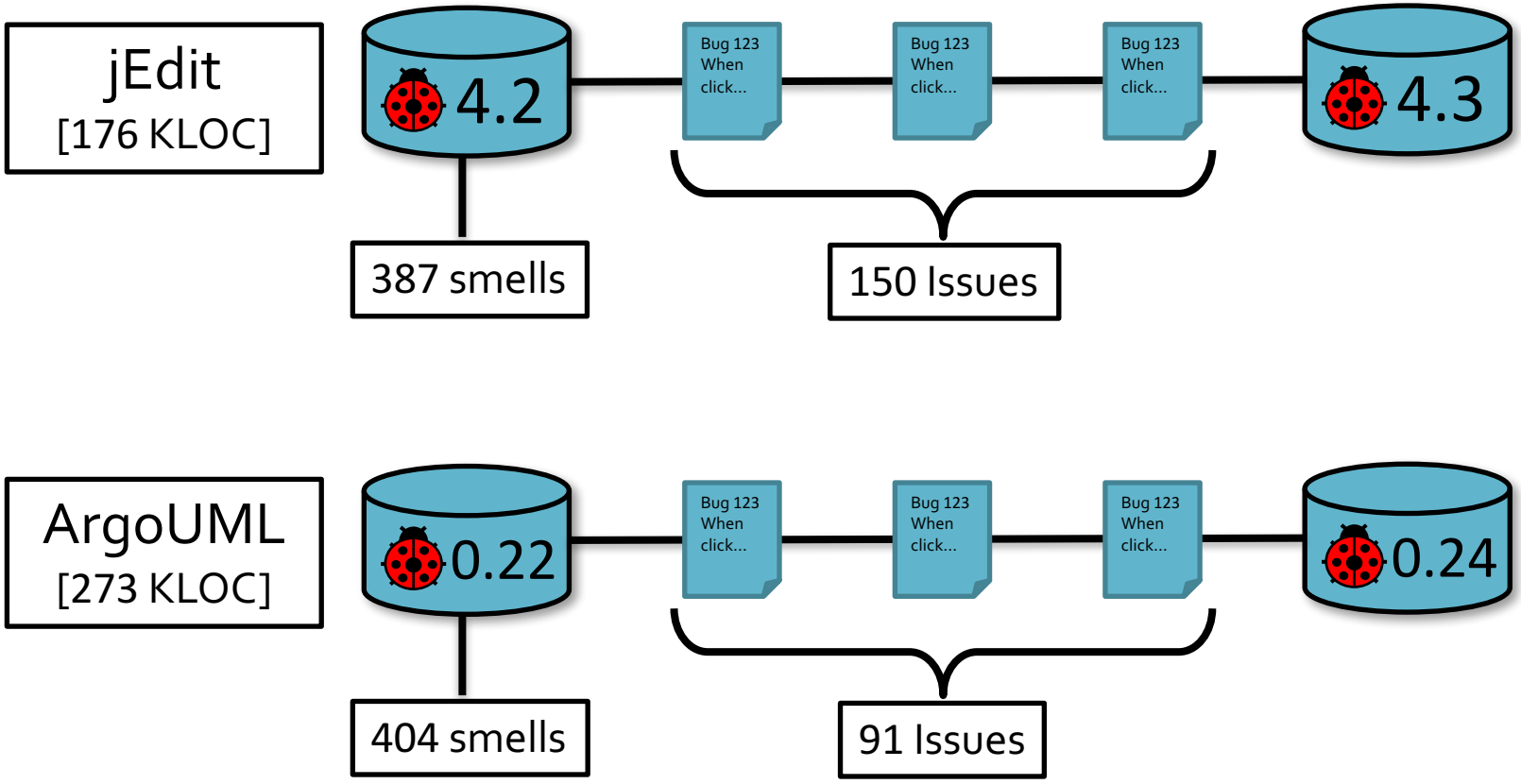
PRELIMINARY EVALUATION

Evaluation questions

- **EQ 1: Does our technique place relevant smells in the higher rank?**
- **EQ 2: Does our technique applicable to every smell level?**
- **EQ 3: Which weighting scheme is better?**
  - Treating every smell equally
  - Using the probability value from FL result

Subjects

- Use Dit et al.'s Feature location benchmark data

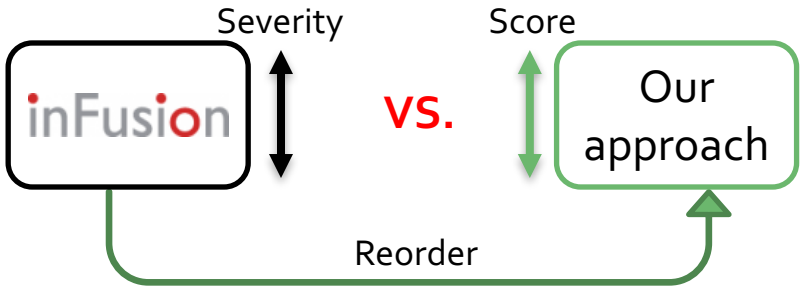


Evaluation metric

Average precision

- Metric for evaluating the quality of ranking documents
- Relevant documents in **higher rank** contribute **more value** than the ones in lower rank

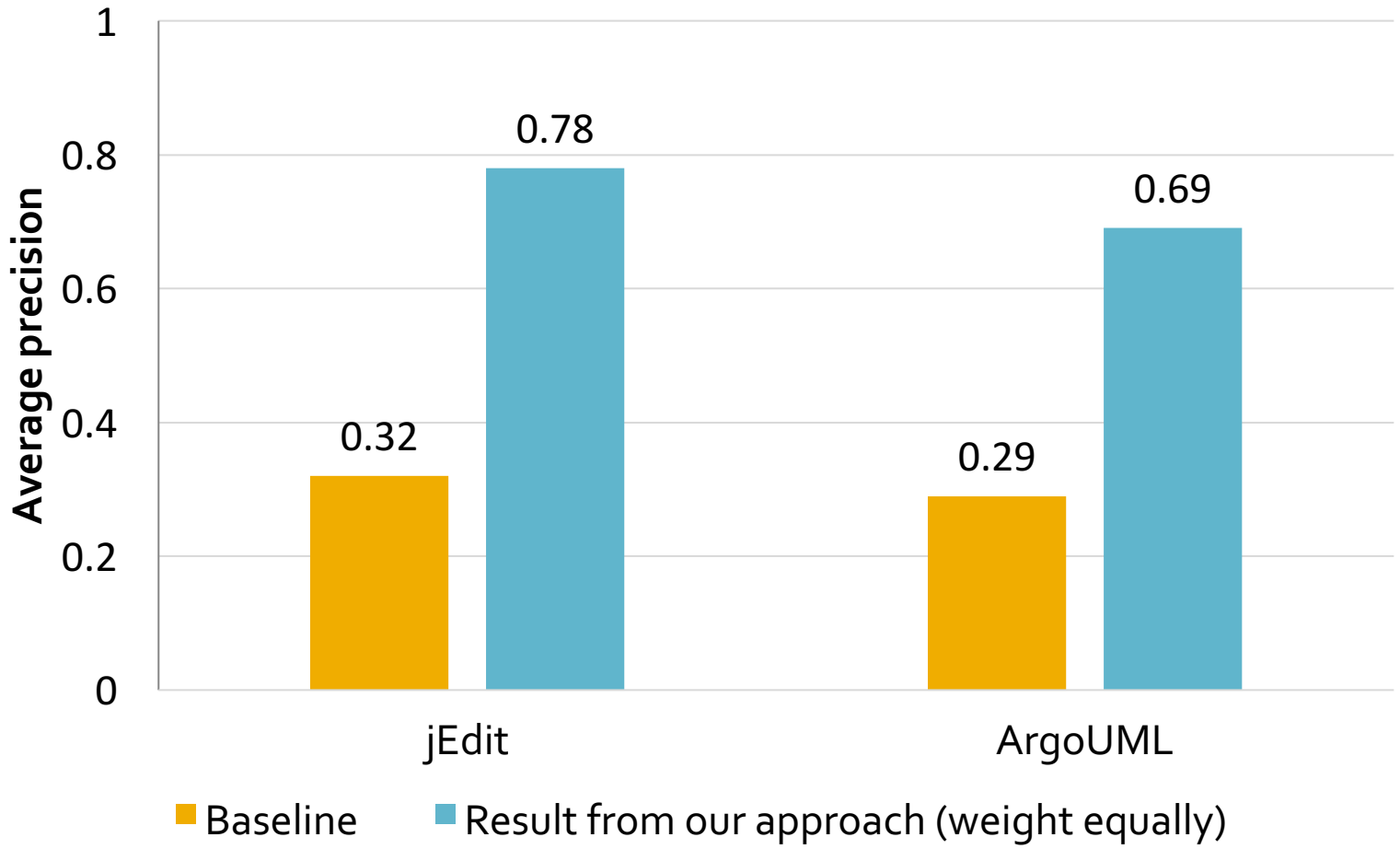
Calculate average precision for:



Gold set

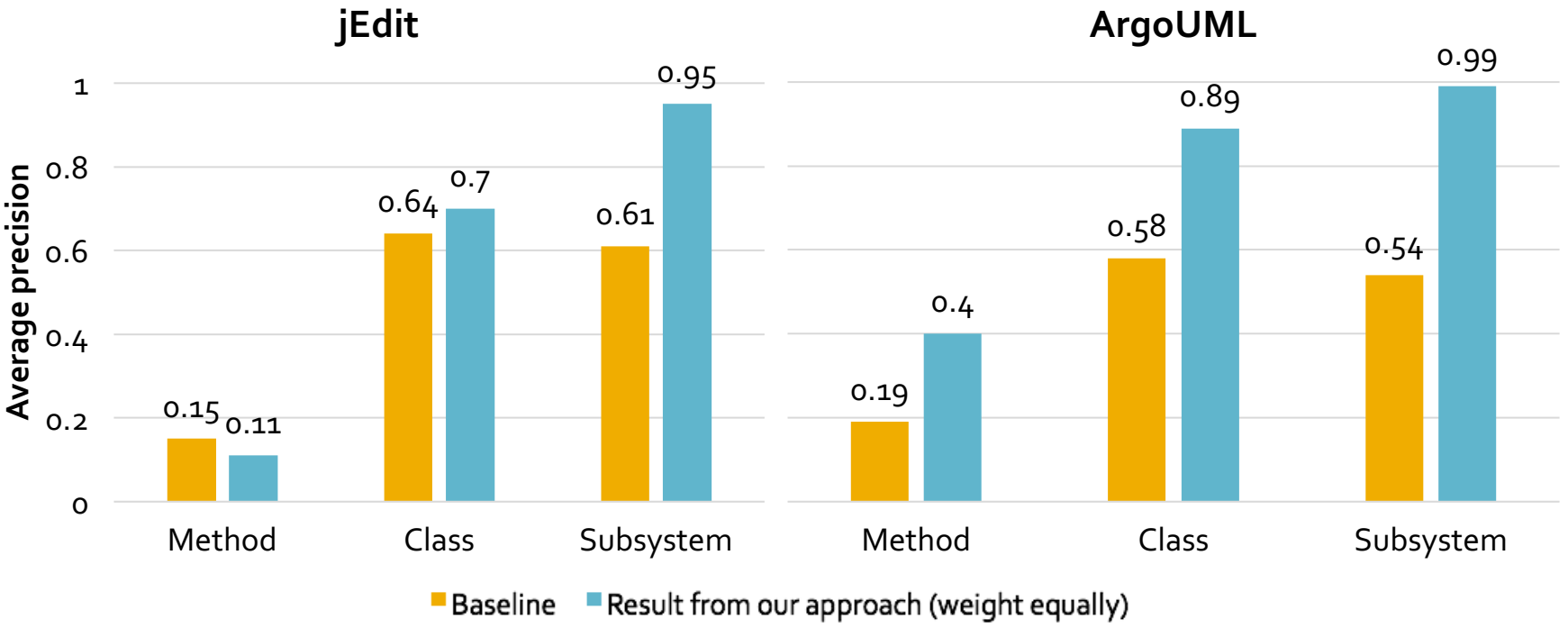
- Smells occurring in the modules *actually* modified during two releases

EQ 1: Are relevant smells in higher rank ?



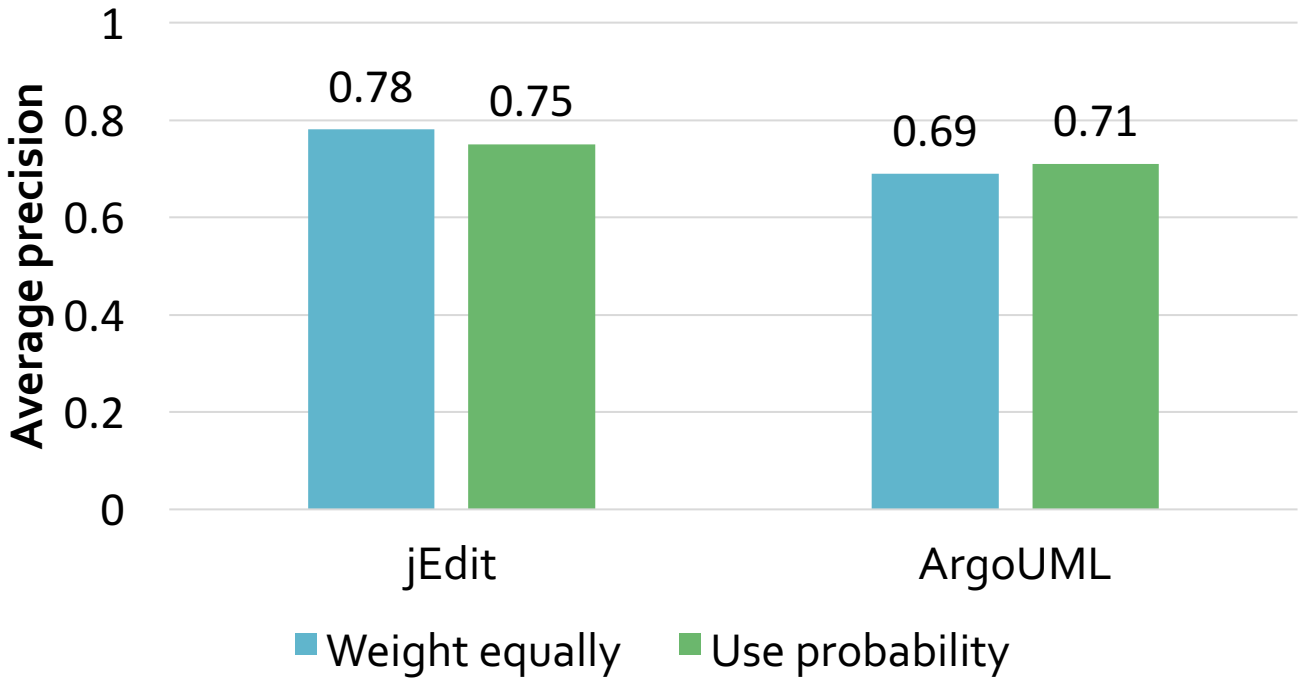
- Yes, according to the average precision value

EQ 2: Applicable to every smell level?



- Our technique is more appropriate with the coarse-grained level code smells

EQ 3 : Weight equally or use probability ?



- **No significant difference**
- **Focus on weighting every smells equally**
  - Simplicity
  - Availability of 'Probability' value from FL technique

Evaluation Questions

- **EQ 1: Are relevant smells in higher rank ?**
  - Yes
- **EQ 2: Applicable to every smell level?**
  - More appropriate with the coarse-grained level
- **EQ 3 : Weight equally or use probability ?**
  - No difference
- **Our approach is potentially effective, but more investigation is needed**

RELATED WORK

Related work

- **Using context of developer for detecting smell**
  - Hayashi et al. [13]
  - Liu et al. [14]
  - Supporting **postfactoring** phase
- **Reducing the number of code smell result**
  - Komatsuda et al. [15]
    - Specifying relevant smell by inserting dummy code
  - Fontana et al. [16]
    - Applying strong and weak filter
  - **Limited** to specific type of smells

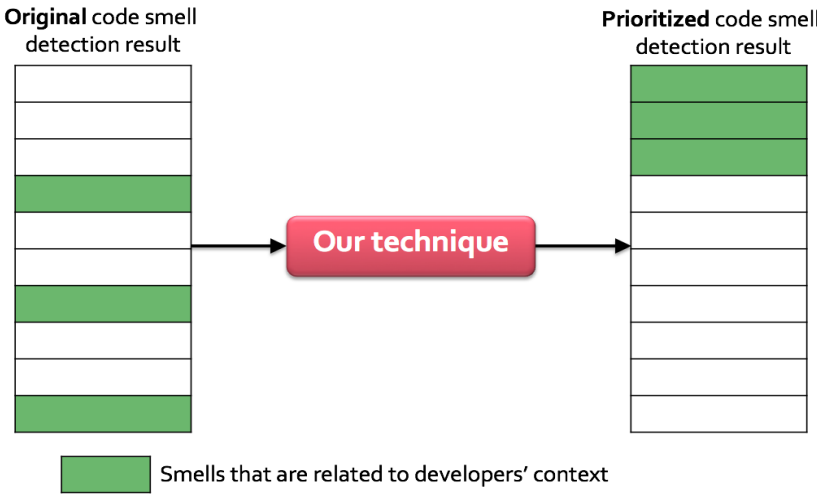
[13] S. Hayashi, M. Saeki, and M. Kurihara, "Supporting refactoring activities using histories of program modification", IEICE2006
[14] H. Liu, X. Guo, and W. Shao, "Monitor-based instant software refactoring", TSE2013
[15] T. Komatsuda, S. Hayashi, and M. Saeki, "Supporting prefactoring using feature location results",IEICE2012
[16] F.A. Fontana, V. Ferme, and M. Zanoni, "Filtering code smells detection results", ICSE2015

FUTURE WORK & CONCLUSION

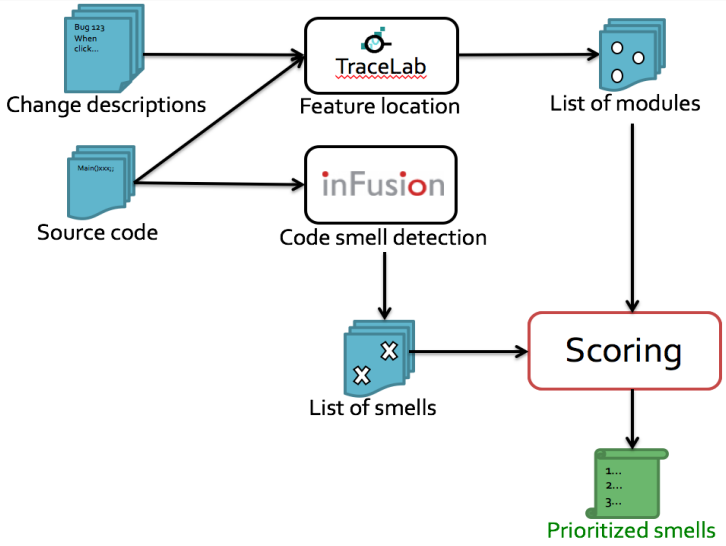
Future work

- **To conduct case studies to confirm that relevant code smells are useful to developers**
- **To consider other factors**
  - The severity of smells
  - The cost needed to fix the smells
  - The importance of the issue

Goal



Approach overview



Scoring

- Counting matched module in FL result

Code smell detection result

Smell	Module	Score
...	...	...
Blob Operation	`ProjectBrowser.loadProject(File,boolean)`	2
...	...	...

Feature location result

#3921

Relevant modules	Prob
`ArgoParser.getProject()`	0.27
`ProjectBrowser.loadProject(File, boolean)`	0.23
...	...
`Project.getBaseName()`	0.04

#4019

Relevant modules	Prob
`ConfigurationHandler.saveDefault()`	0.36
`ProjectBrowser.loadProject(File, boolean)`	0.22
...	...
`UMLToDoItem.select()`	0.03

Evaluation Questions

- EQ 1: Are relevant smells in higher rank ?
- Yes
- EQ 2: applicable to any entity type ?
- More appropriate with the coarse-grained level
- EQ 3 : weight equally or use probability ?
- No difference