CSBSE 2018

代码坏味的检测与重构

刘辉

北京理工大学 计算机学院

2018.11.17

大纲

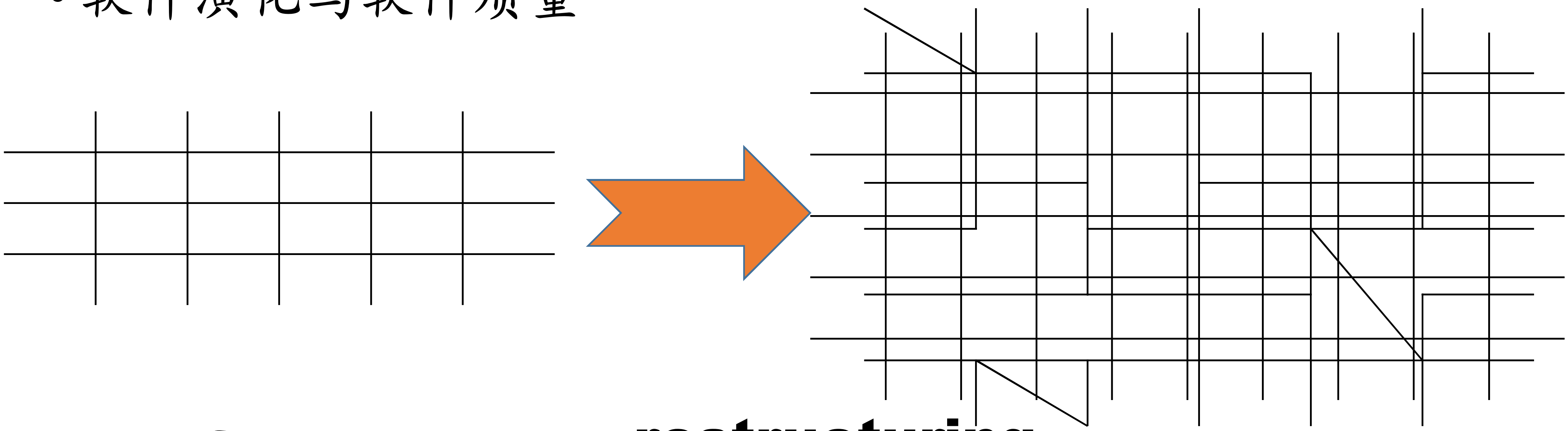
- 代码坏味与软件重构
- 基于监控和反馈的代码坏味检测与重构
- 基于机器学习的坏味检测与重构推荐

大纲

- 代码坏味与软件重构
- 基于监控和反馈的代码坏味检测与重构
- 基于机器学习的坏味检测与重构推荐

软件优化与软件重构

- 软件演化与软件质量

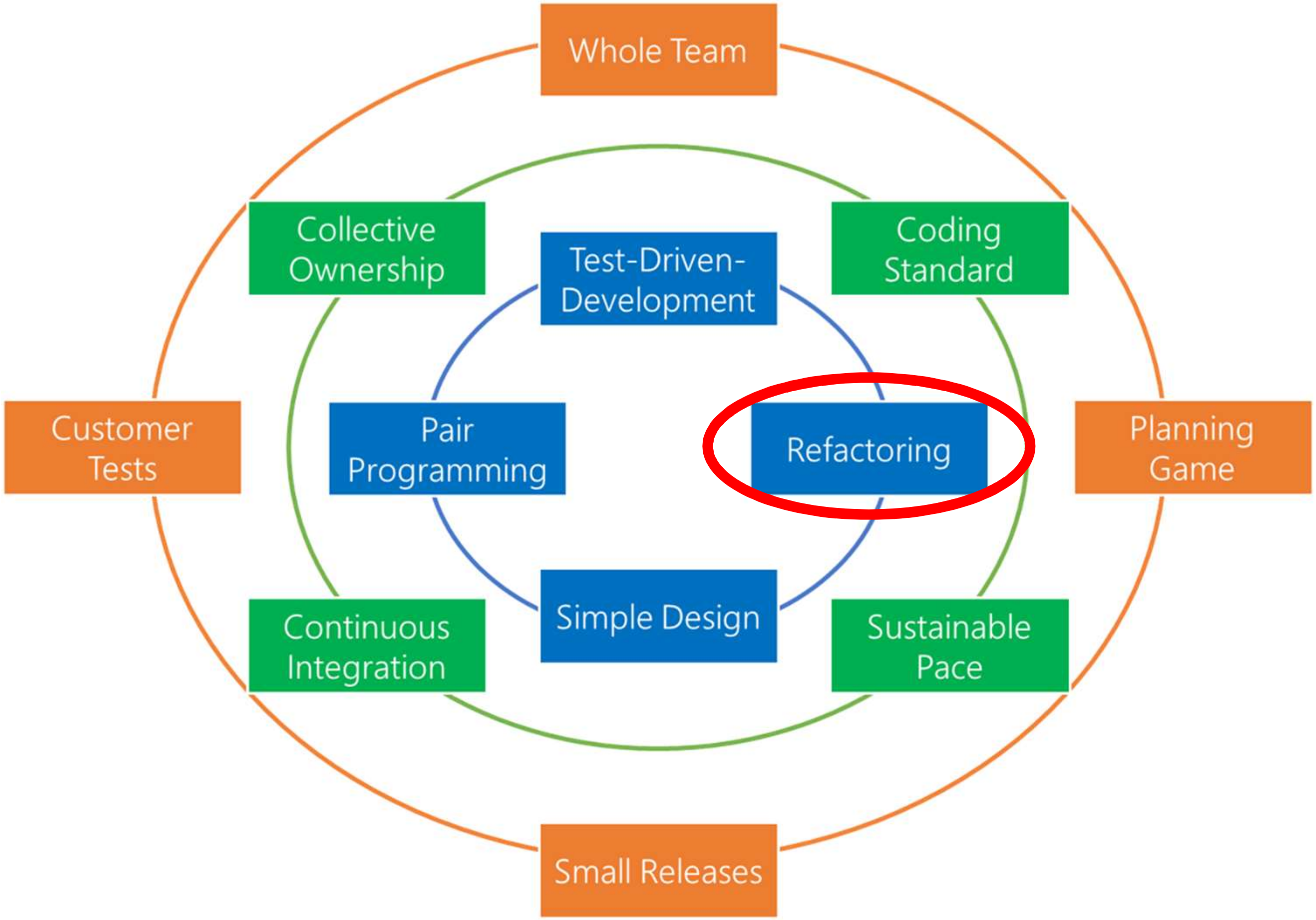


refactoring **restructuring**

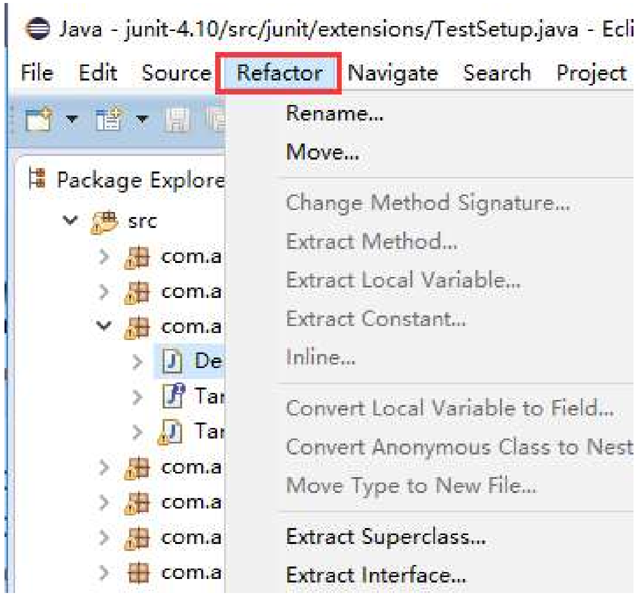
软件重构的主要特征

- 目标
  - 提高代码的可读性、可维护性、可扩展性
- 手段
  - 调整代码内部结构
- 特色
  - 不得改变软件的功能

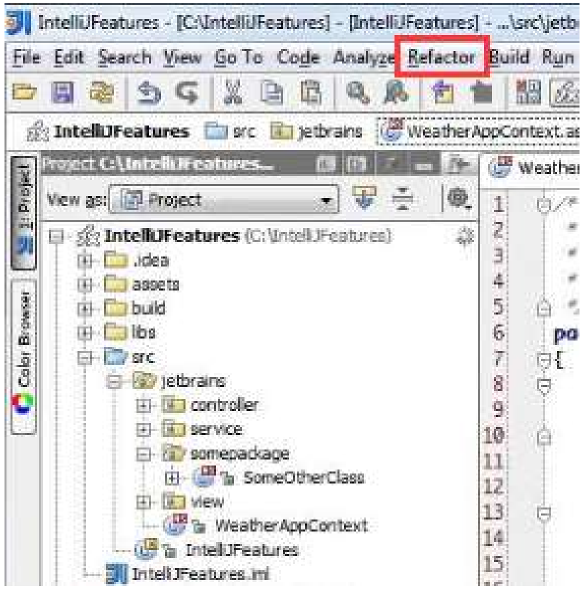
XP与软件重构


Whole Team
Collective Ownership
Test-Driven-Development
Coding Standard
Customer Tests
Pair Programming
Refactoring
Planning Game
Continuous Integration
Simple Design
Sustainable Pace
Small Releases

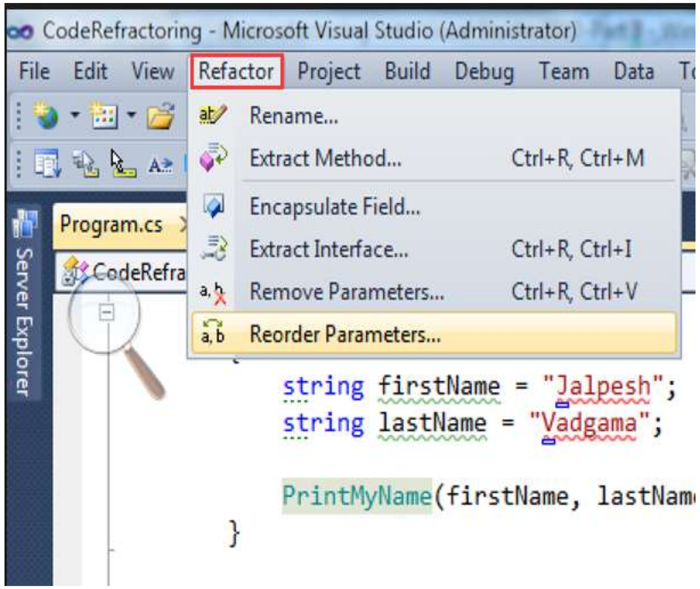
工业应用



Eclipse

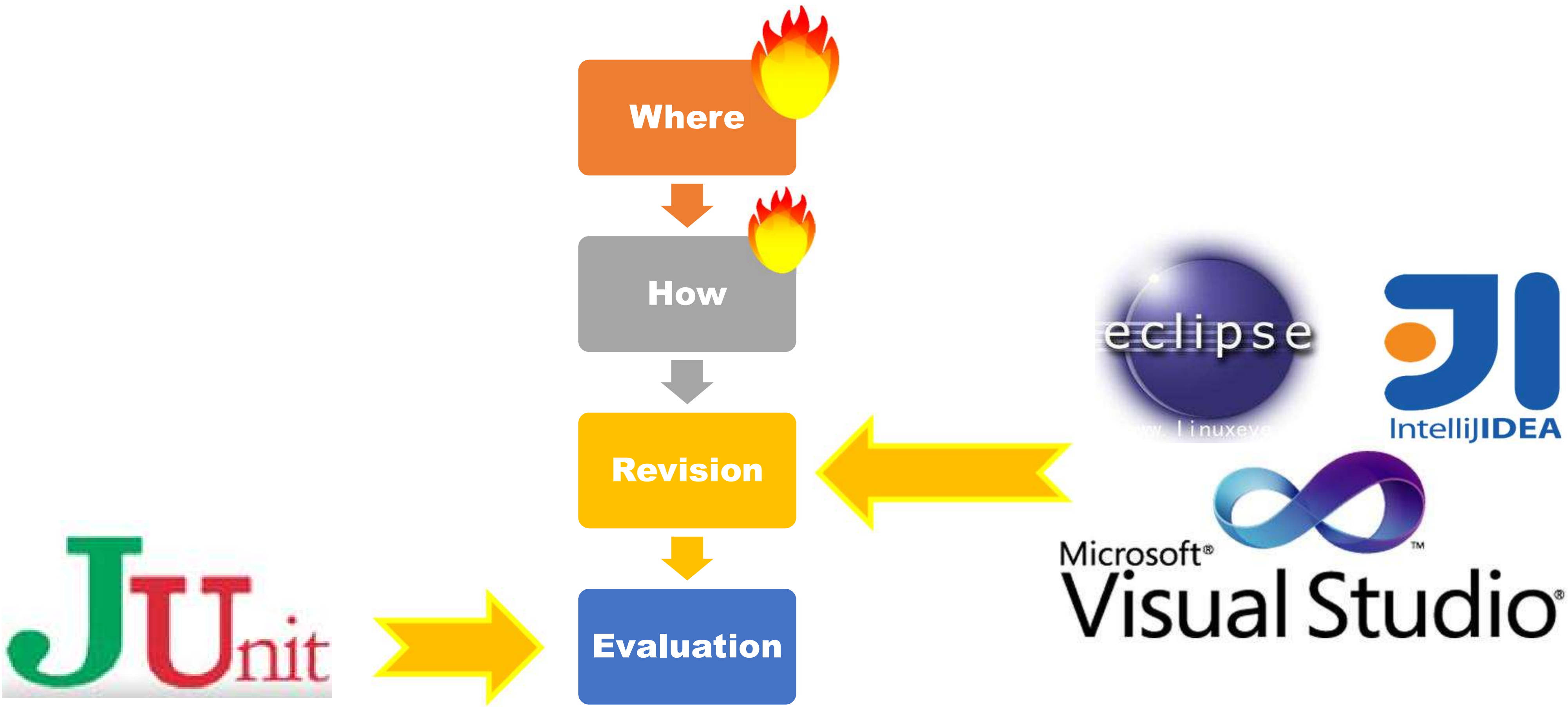


IntelliJ



Visual Studio

软件重构的流程



Where
How
Revision
Evaluation
JUnit
eclipse
IntelliJIDEA
Microsoft® Visual Studio®

Code Smells 与软件重构

- 重复代码/克隆代码 **Duplicated code**
- 上帝类 **Large class/God object** → **Extract Class**
- 特征依恋**Feature envy**
- 过于亲密 **Inappropriate intimacy**
- 拒绝继承 **Refused bequest**（遗产）
- 长方法 **Long Method**

代码坏味与软件度量

- **God Object**

A god object is an object that knows too much or does too much.

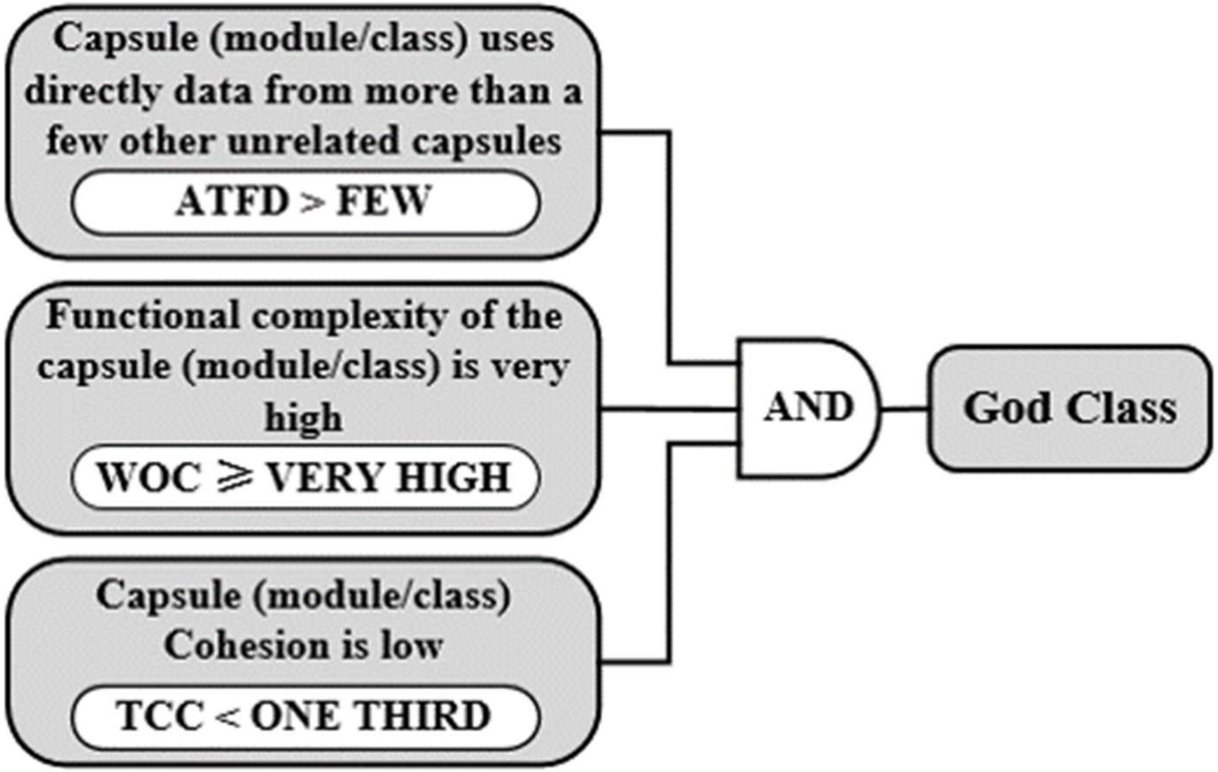
代码坏味与软件度量

- **God class**

ATFD: Access to Foreign Data

WOC: Weighted Operation Count

TCC: Tight Capsule Cohesion



代码坏味与软件度量

Long live code smells!

代码坏味与软件度量

- 关键挑战
  - 度量本身有效性 软件度量是否反映*smell*的内涵？
    - 度量值能否准确反映软件的质量？缺乏语义分析
    - 不同的人可能采用不同的度量检测相同的*smell*
      - *Long method*
  - 重构的主观性和上下文 重构推荐是否需要因人/项目而异？
  - 软件重构的效益 重构是否值得？
    - 重构本身不会带来新的功能（收益），不会对最终用户产生直接影响。
      - 高风险、高成本的软件重构是为了度量？
      - *Clone*

大纲

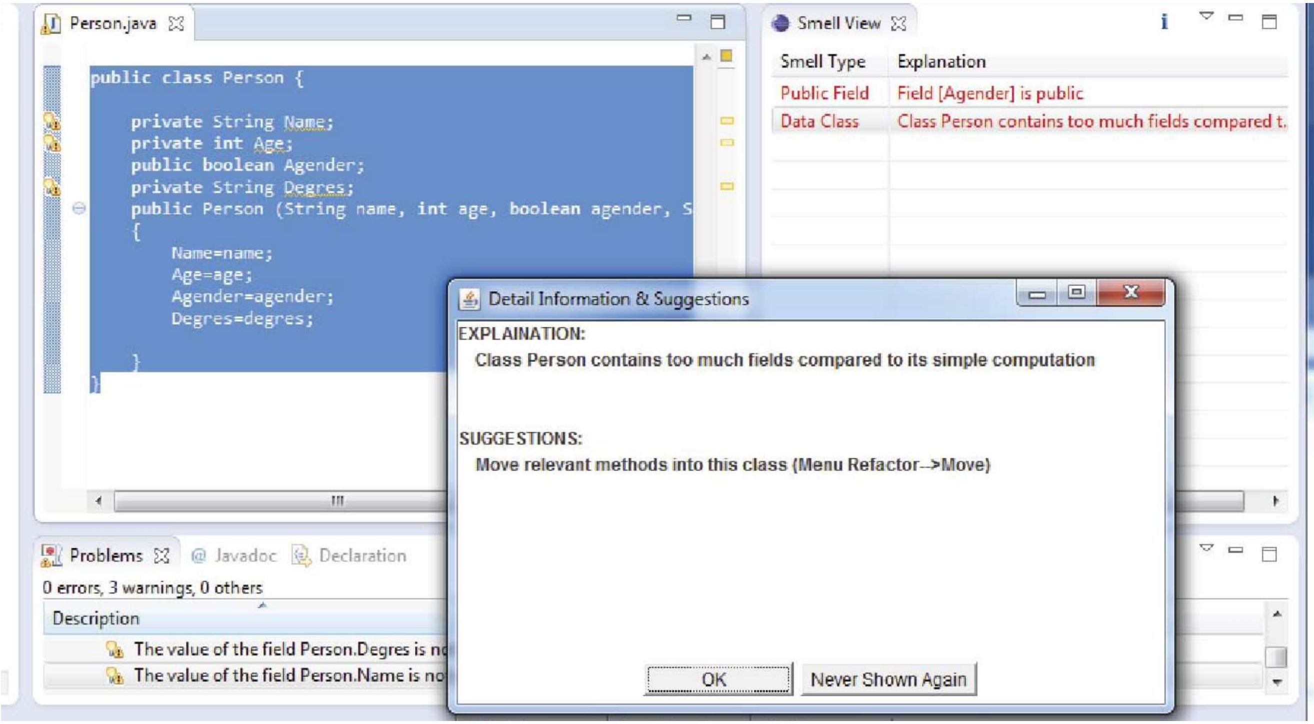
- 代码坏味与软件重构
- 基于监控和反馈的代码坏味检测与重构
- 基于机器学习的坏味检测与重构推荐

基于监控和反馈的重构推荐

- **要解决的问题**
  - **依赖程序员的主动性**
    - **未解决、迟检测**
  - **检测速度**
    - **非增量式**
  - **缺乏交互**
    - **缺乏个性化**

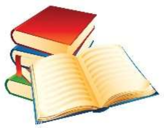
基于监控和反馈的重构推荐

- 基于监控的增量式**smell**检测与重构
  - *增量式smell检测*
  - *重构方案的即时推荐*
  - *重构识别*



基于监控和反馈的重构推荐

- 降低重构代价，**Smell**生存期降低（**92%**）
- 解决了更多的**code smell** （**2.4**倍）
- 减少**smell**出现概率（**51%**）

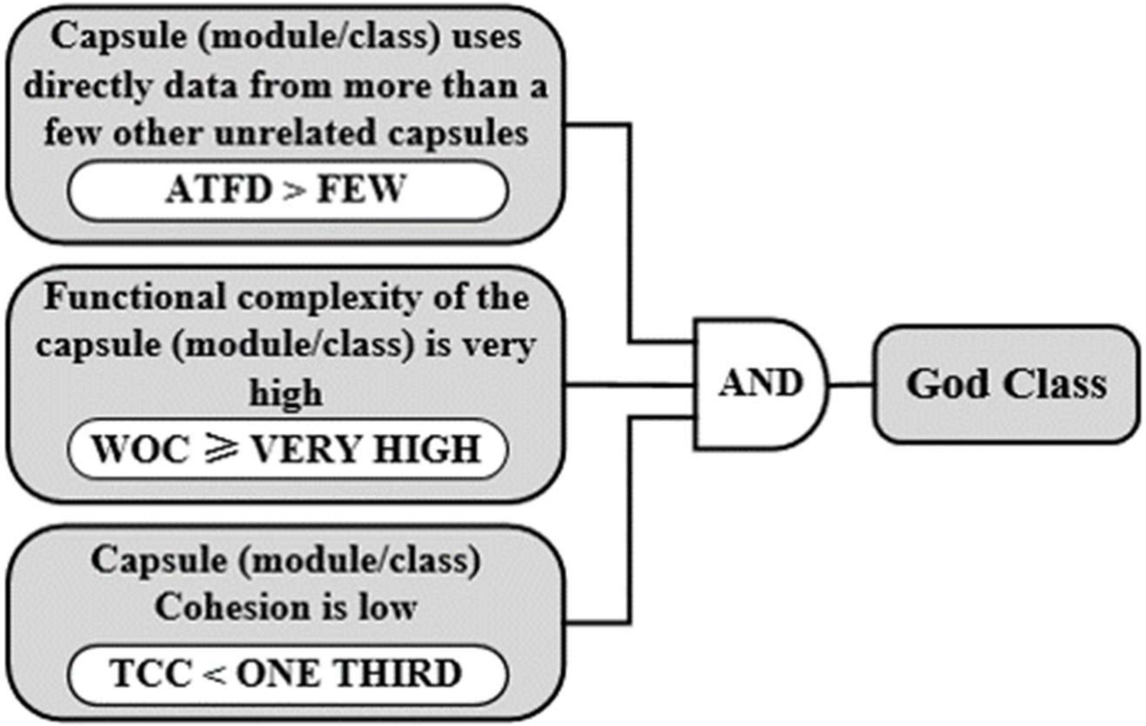


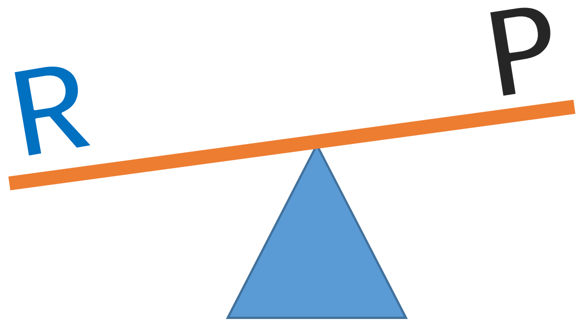
HUI LIU et al. *Monitor-based Instant Software Refactoring*, IEEE TSE 2013

基于监控和反馈的重构推荐

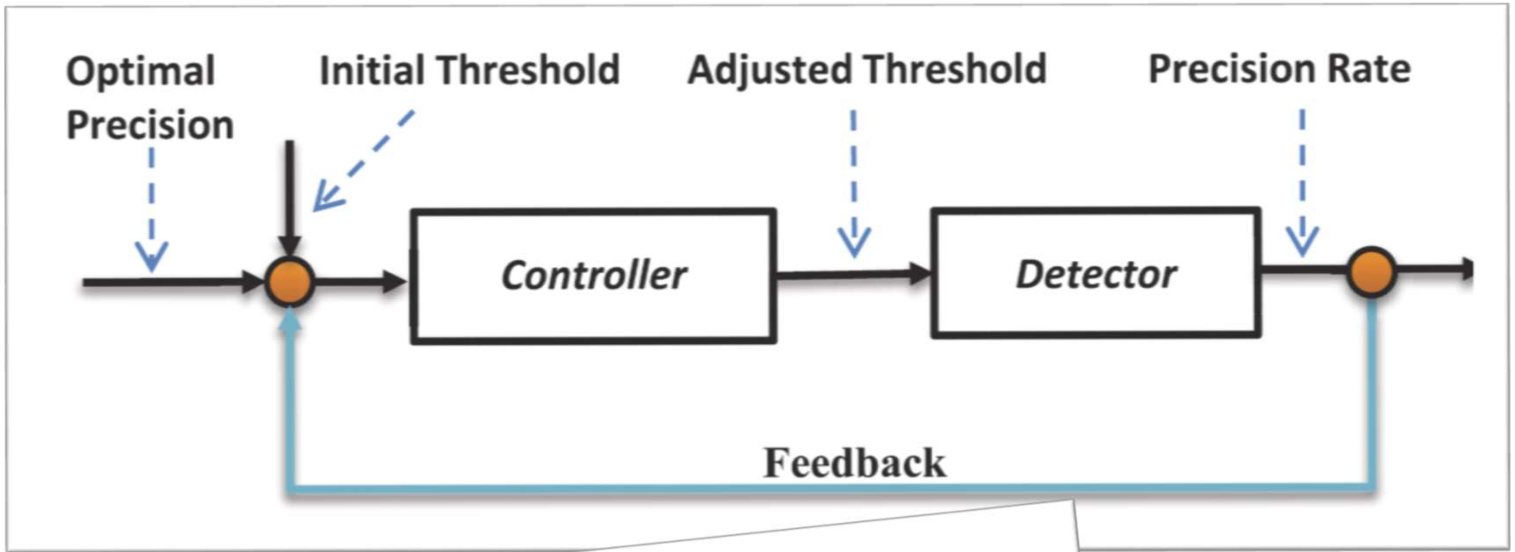
- 要解决的问题
  - ~~依赖程序员的主动性~~
    - ~~未解决、迟检测~~
  - ~~检测速度~~
    - ~~非增量式~~
  - **缺乏交互**
    - 缺乏个性化

基于监控和反馈的重构推荐





基于监控和反馈的重构推荐



$$
\begin{aligned}
&\max r \\
&\text{subject to} \\
&\qquad p \geq targetP,
\end{aligned}
$$

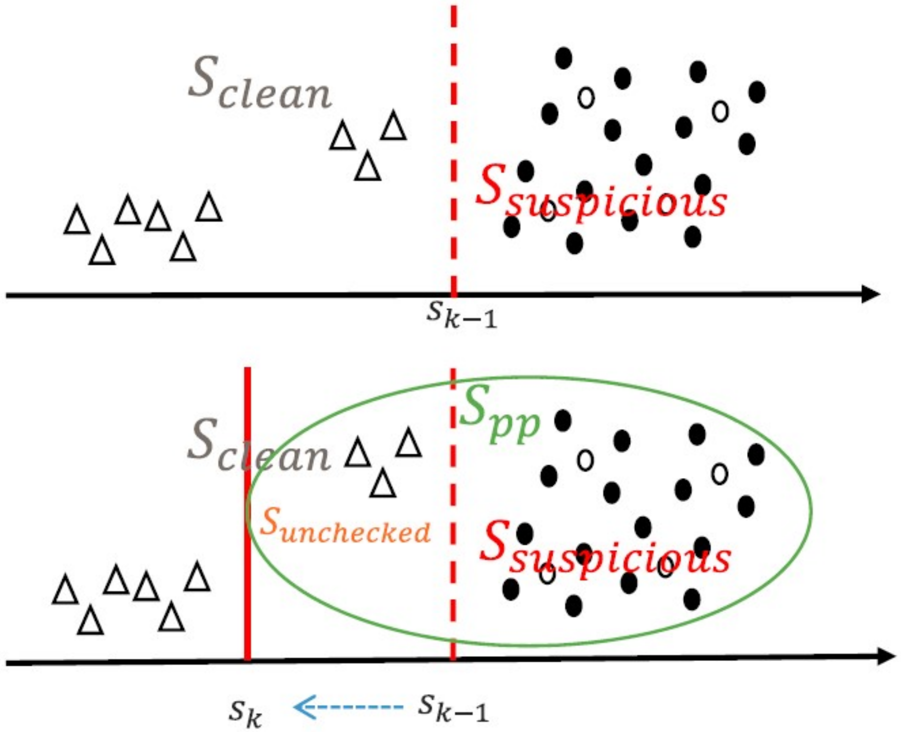
基于监控和反馈的重构推荐

$$\begin{aligned} &\max r \\ &\text{subject to} \\ &\qquad p \geq targetP, \end{aligned}$$

$$\begin{aligned} &\max_s tp(s, dataSet) \\ &\qquad \text{subject to} \\ &\qquad\qquad p(s, dataSet) \geq targetP. \end{aligned}$$

$$f(s,dataSet) = \begin{cases} tp(s, dataSet) & \text{if } p(s, dataSet) \geq targetP \\ p(s, dataSet) & \text{others.} \end{cases}$$

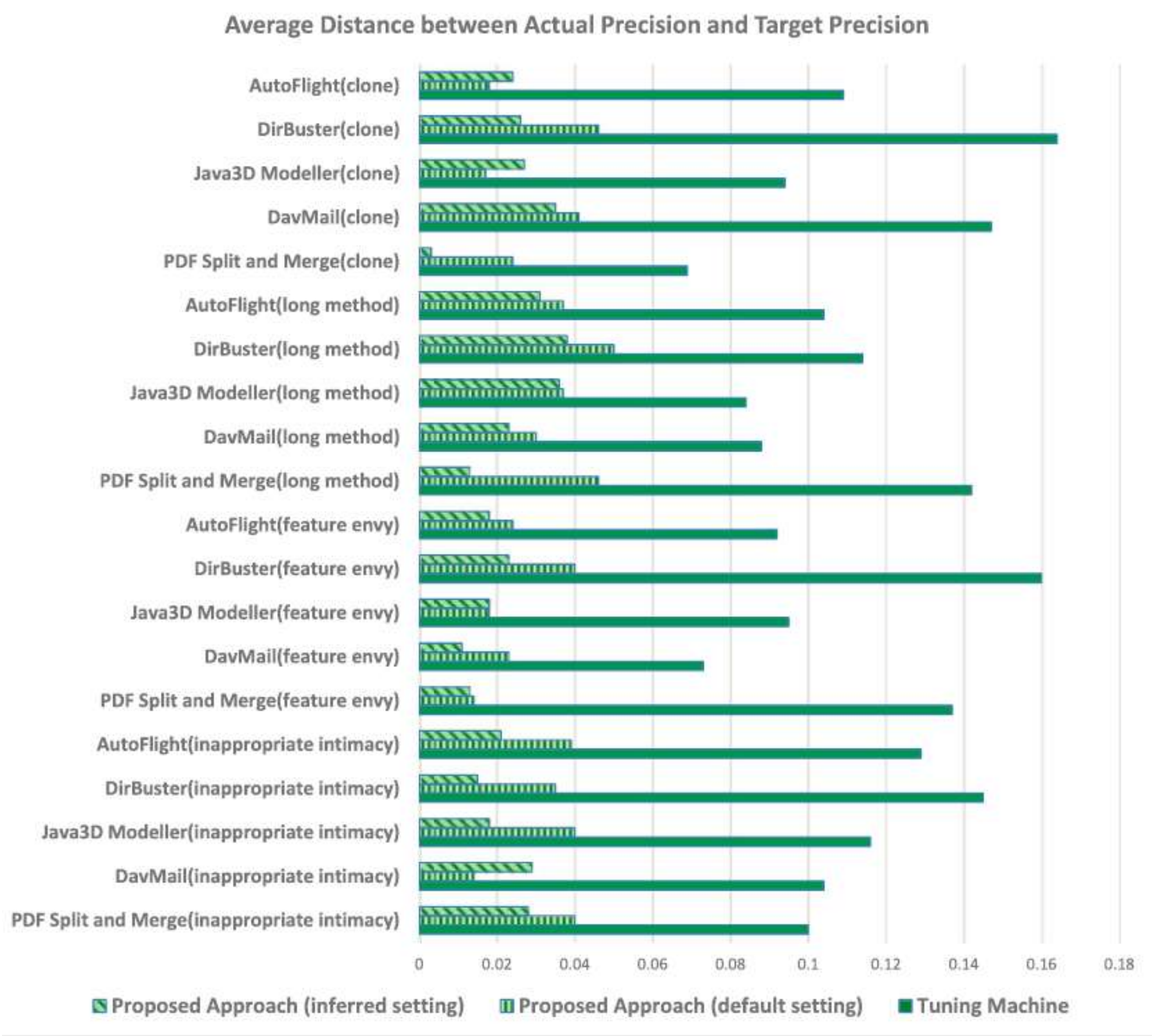
基于IDE监控与反馈的阈值优化



$$p(s_k, Sunchecked) = p(s_k, Schecked) \times e^{-\frac{|Sunchecked|}{|Schecked|}}$$

- 误差降低

一个数量级



HUI LIU et al. ***Dynamic and Automatic Feedback-Based Threshold Adaptation for Code Smells Detection*****, IEEE TSE 2016**

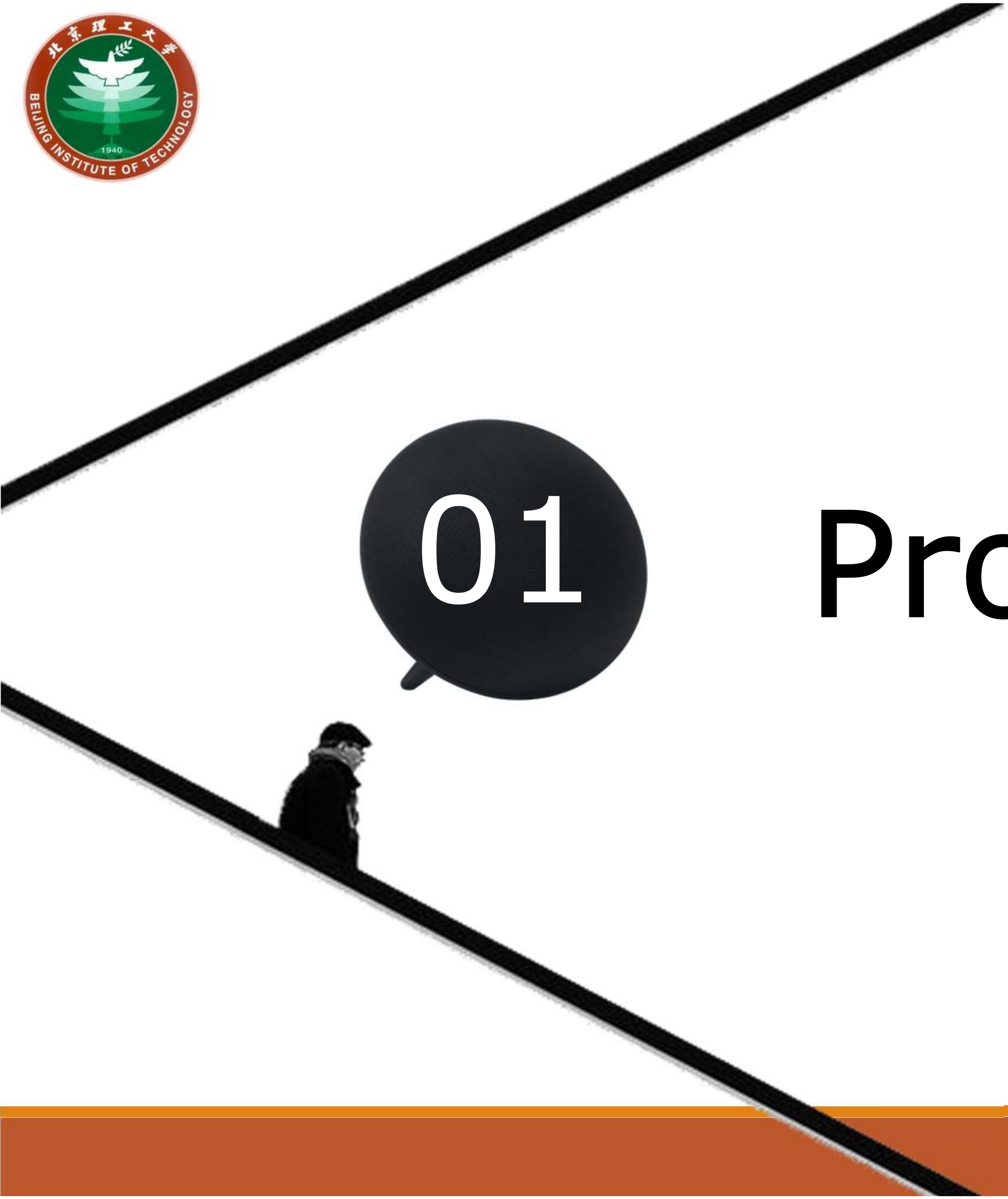
大纲

- 代码坏味与软件重构
- 基于监控和反馈的代码坏味检测与重构
- 基于机器学习的坏味检测与重构推荐

Deep Learning Based Feature Envy Detection

HUI LIU et al. *Deep Learning Based Feature Envy Detection*, ASE 2018

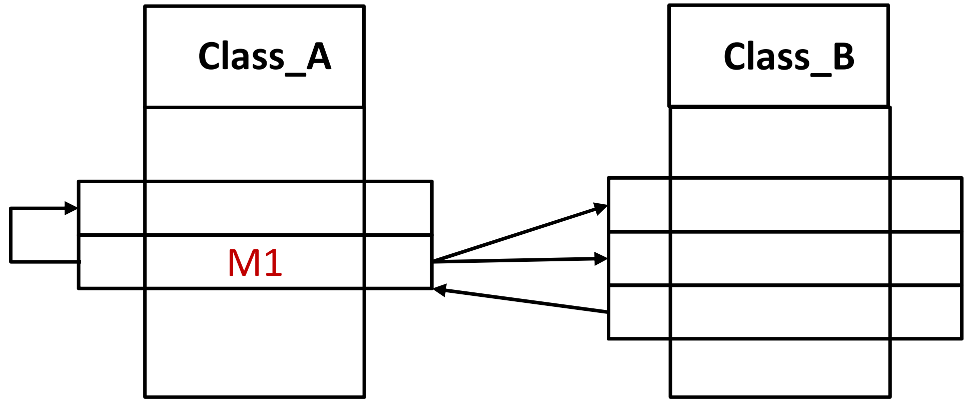
CONTENTS



01 Problem

Feature Envy

A method is more interested in another class than the class where it is.



Detection and Resolution

Existing approach to detect feature envy :

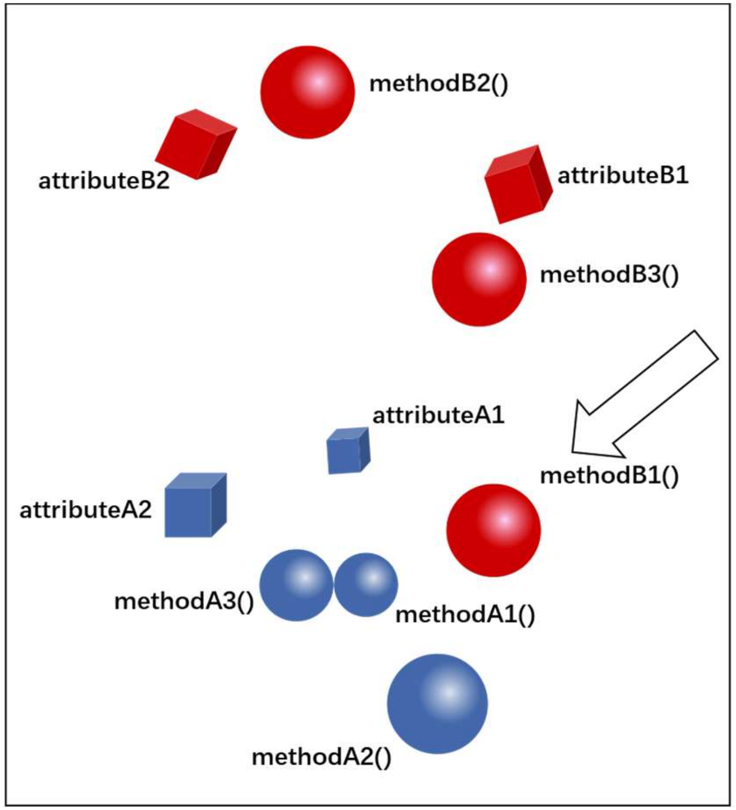
- Metrics Based
- Textual Information Based
- Change History Based
- Refactoring History Based

F. Simon et al. Metrics Based Refactoring. CSMR 2001

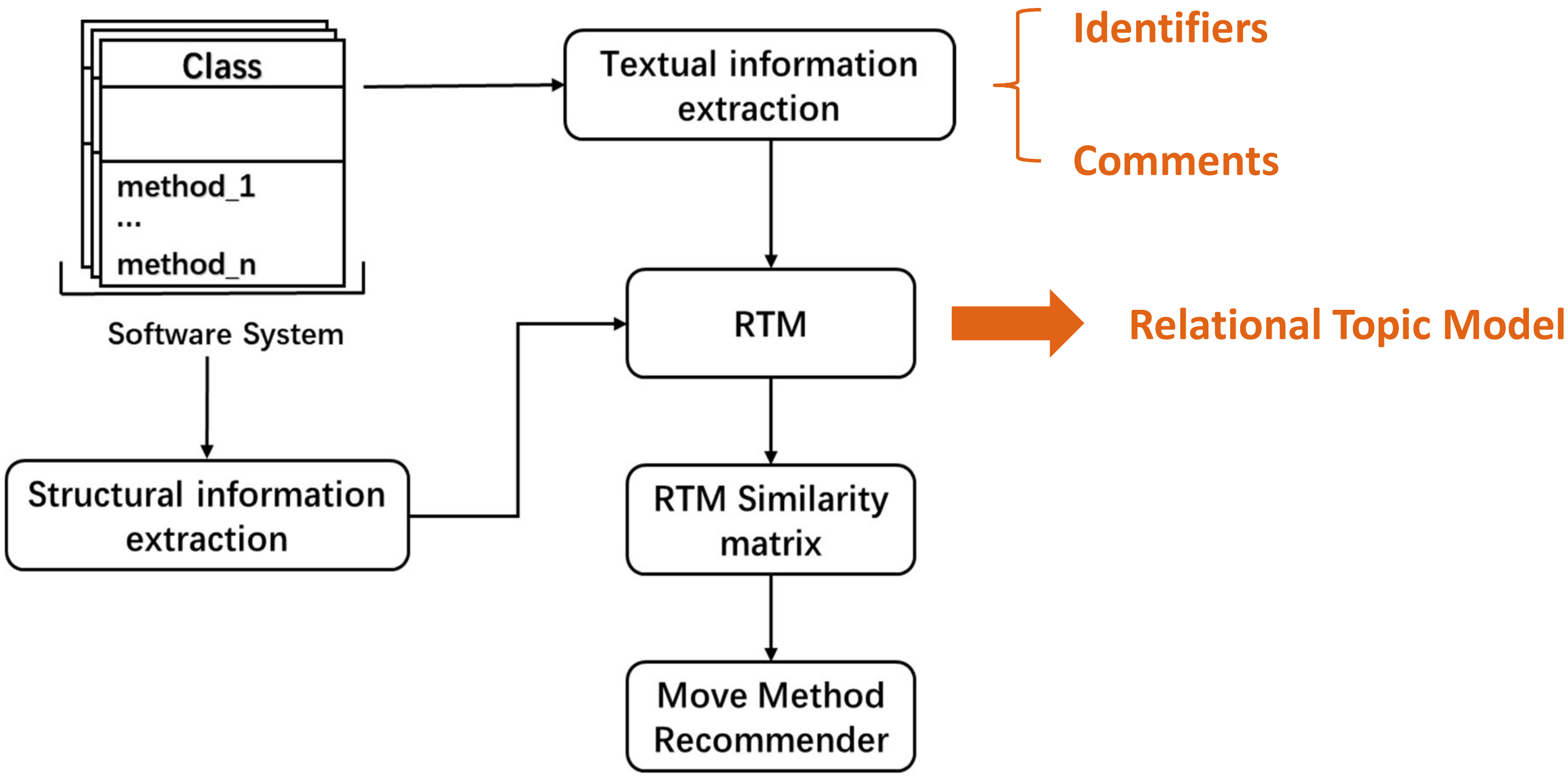
Metrics Based Approach

- $$distance(e_1, e_2) = 1 - \frac{|p(e_1) \cap p(e_2)|}{|p(e_1) \cup p(e_2)|}$$

- $$distance(m, C) = 1 - \frac{S_m \cap SC}{S_m \cup SC}$$

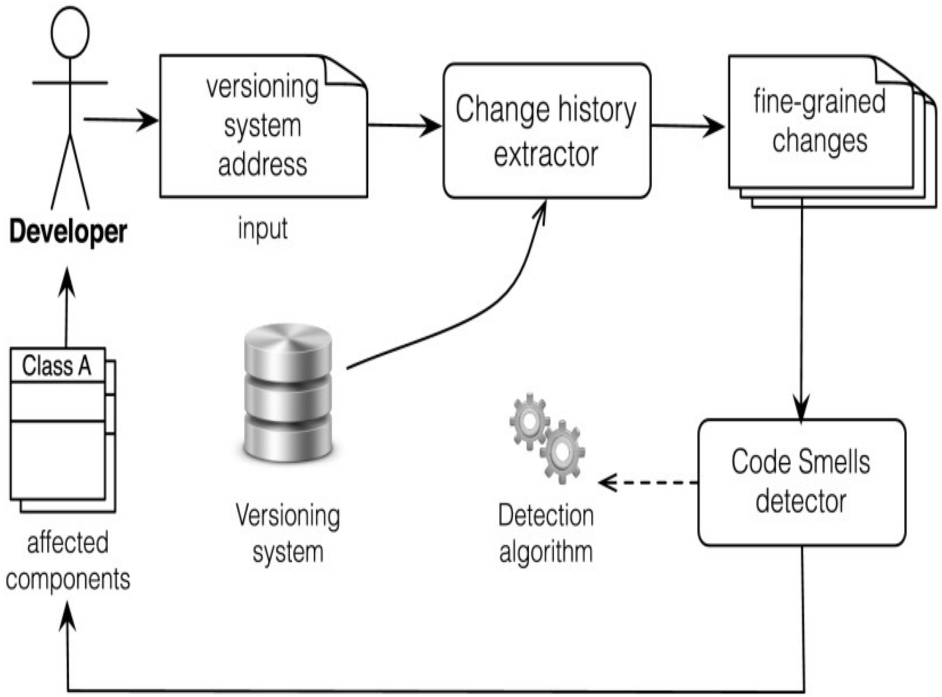


Textual Information Based Approach



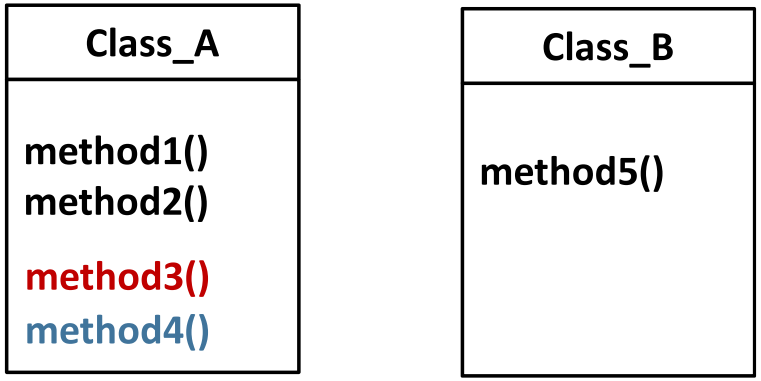
Change History Based Approach

A method affected by feature envy changes more often with the envied class than with the original class.



Refactoring History based Approach

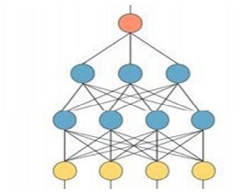
Similar and closely related methods should be moved together.



H. Liu *et al.*. Domino Effect: Move More Methods Once a Method is Moved. SANER 2016

Detection and Resolution

NONE of them exploits
deep learning !



Deep Learning

Achieve great advances recently

Very powerful

Select features automatically

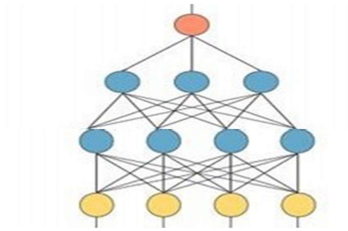
Mapping input to output

Why not exploits deep learning?

Detection and Resolution

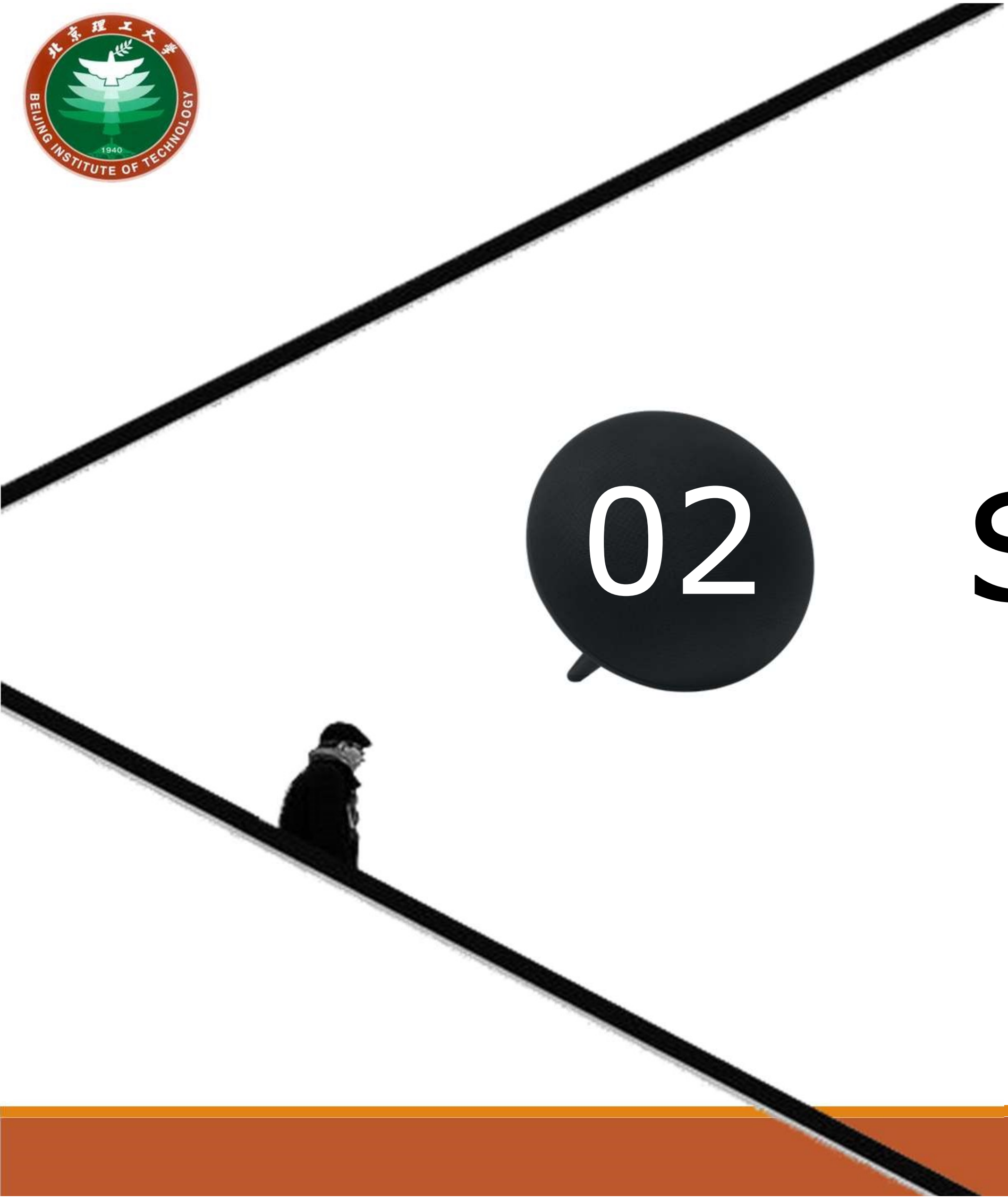
Lack of large labeled training data

Manually built



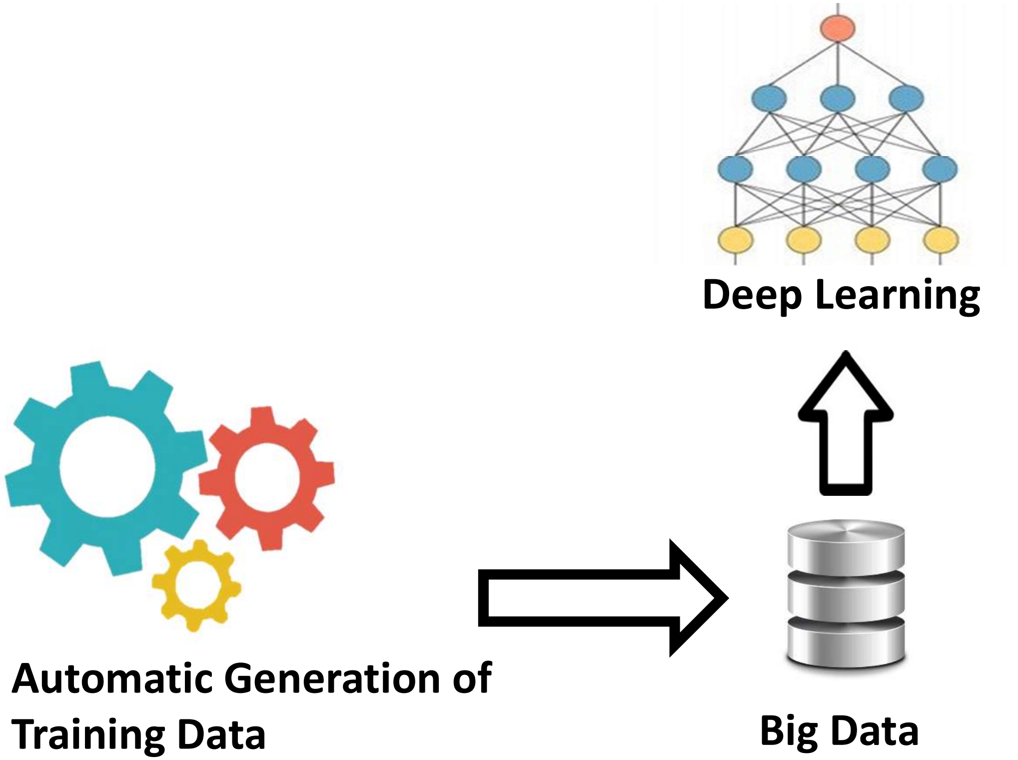
Deep Learning

Big Data

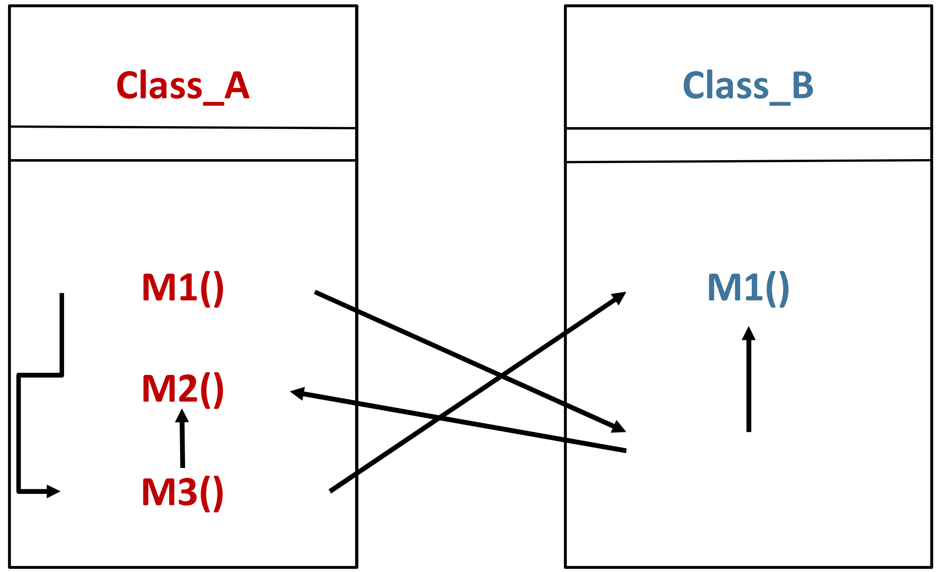


02 Solution

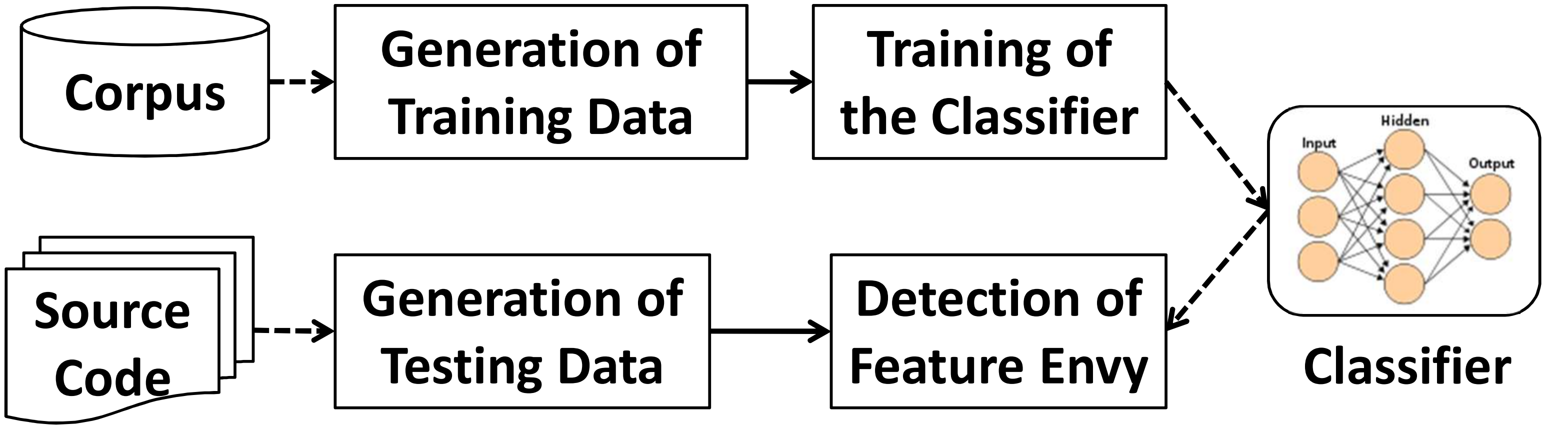
Generation of Training Data



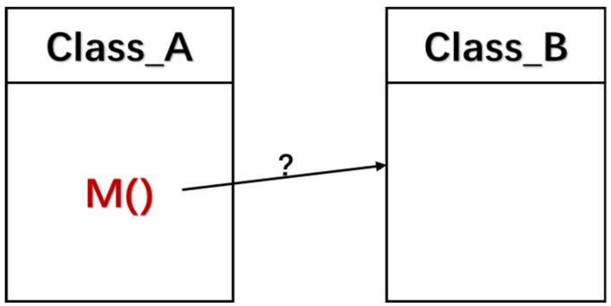
Generation of Training Data

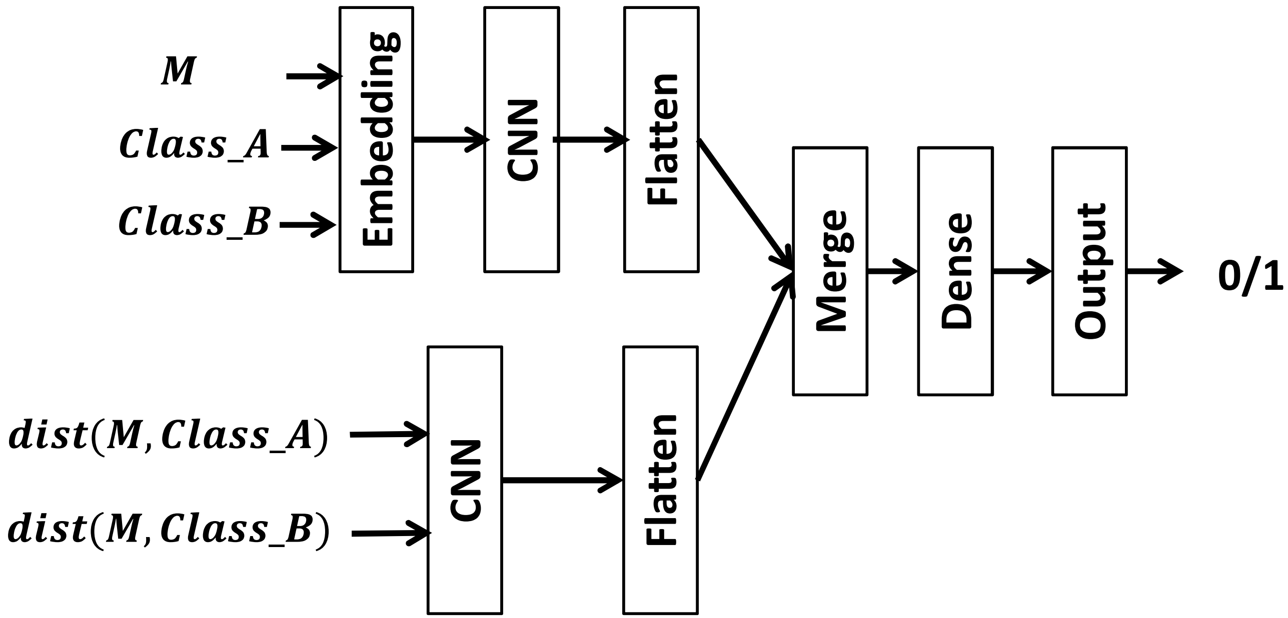


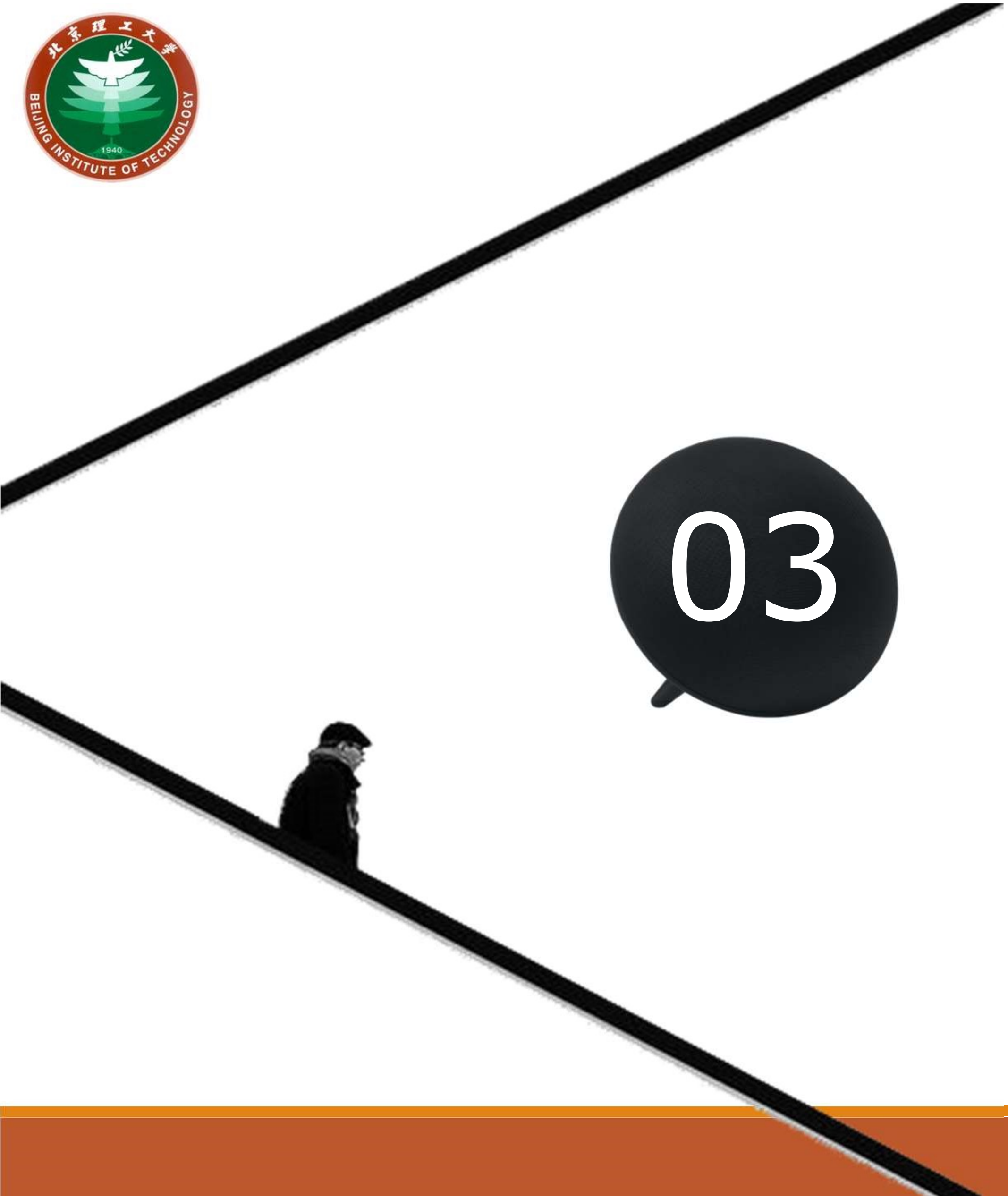
Overview


Corpus
Generation of Training Data
Training of the Classifier
Source Code
Generation of Testing Data
Detection of Feature Envy
Hidden
Input
Output
Classifier

Deep Learning Based Feature Envy Detection







03 Evaluation

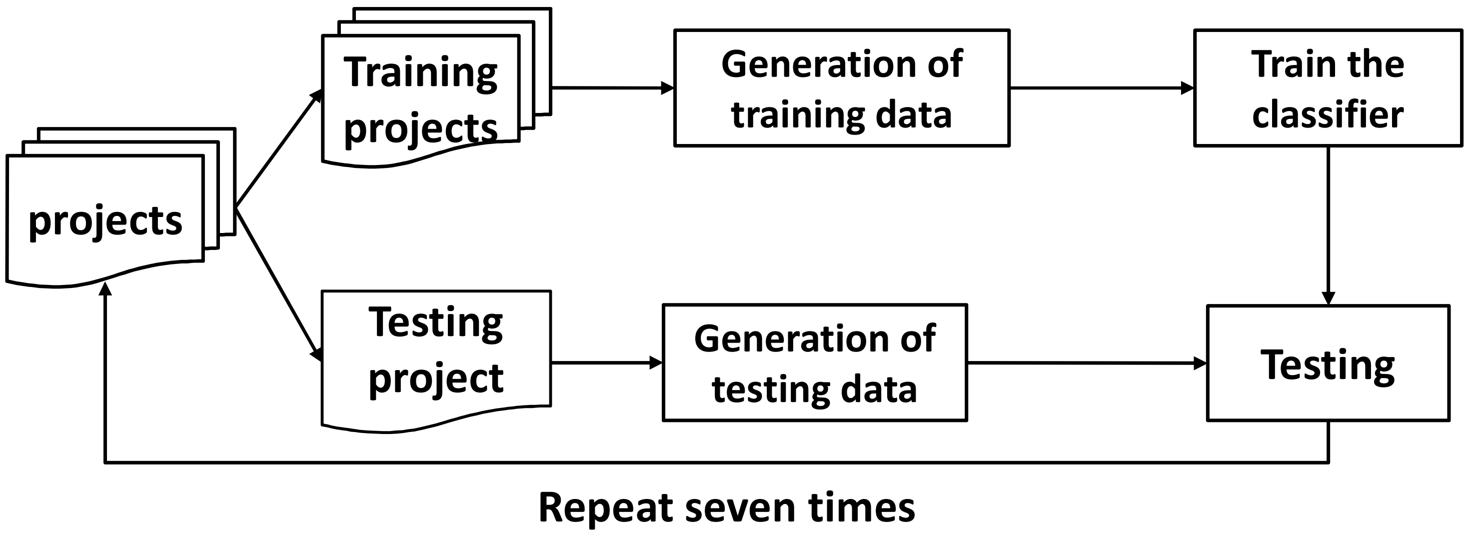
Research Questions

- RQ1: Outperform the state-of-the-art approaches?
- RQ2: Accurate of recommending destinations (where to move) ?
- RQ3: How does textual input / code metrics influence the performance?

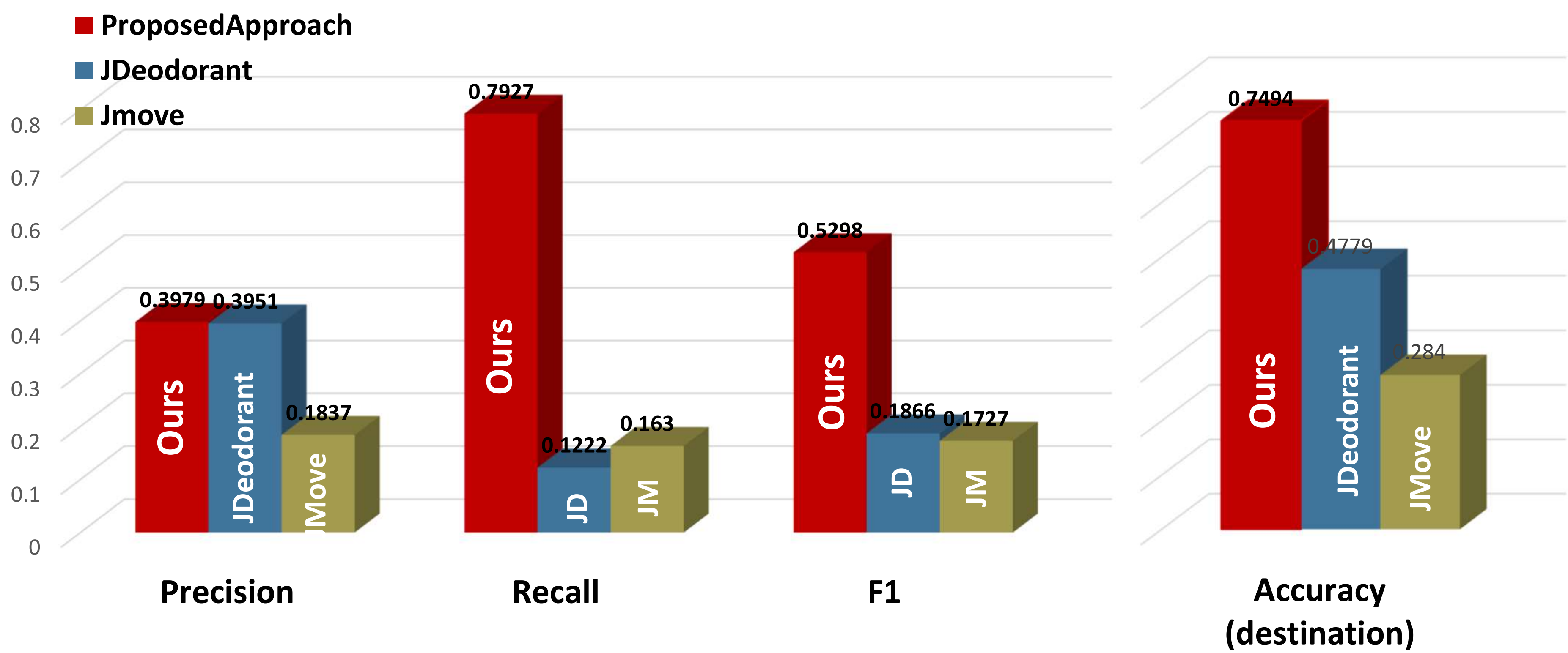
Process

Applications	Domain	#Classes	#Methods	LOC
JUnit	unit testing	123	866	11,734
PMD	static code analysis	250	2,097	32,783
JExcelAPI	Excel API	424	3,118	90,555
Areca	file backup	473	5,055	88,126
Freeplane	knowledge management	787	6,938	124,937
jEdit	text editor	513	5,964	185,571
Weka	machine learning	1348	20,182	444,493

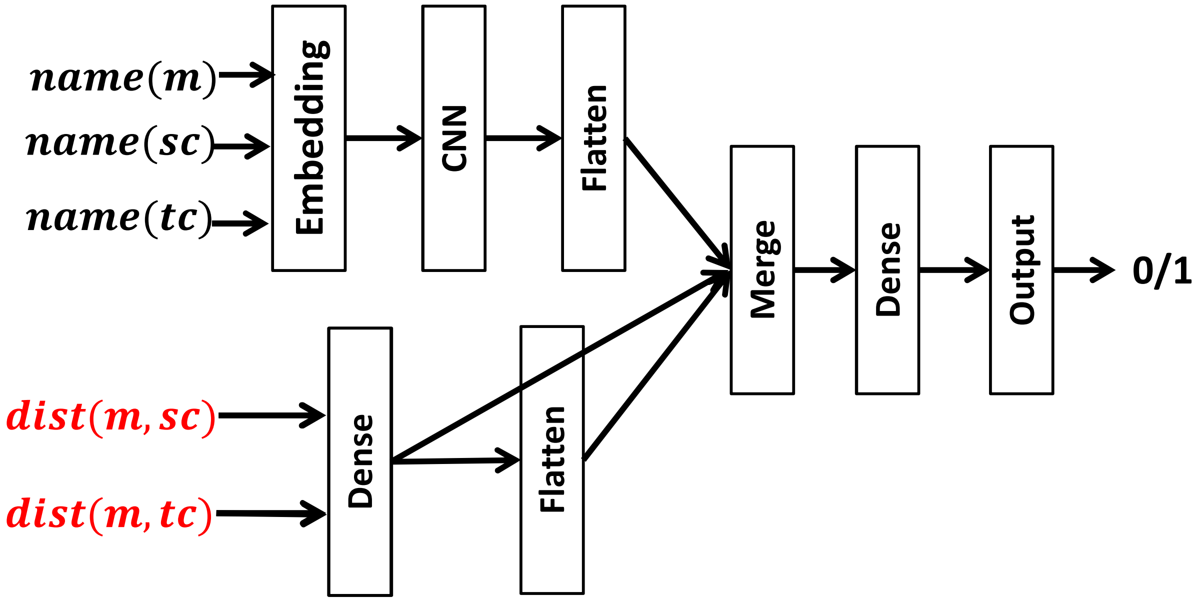
Process


projects
Training projects
Testing project
Generation of training data
Generation of testing data
Train the classifier
Testing
Repeat seven times

Improve the State of the art

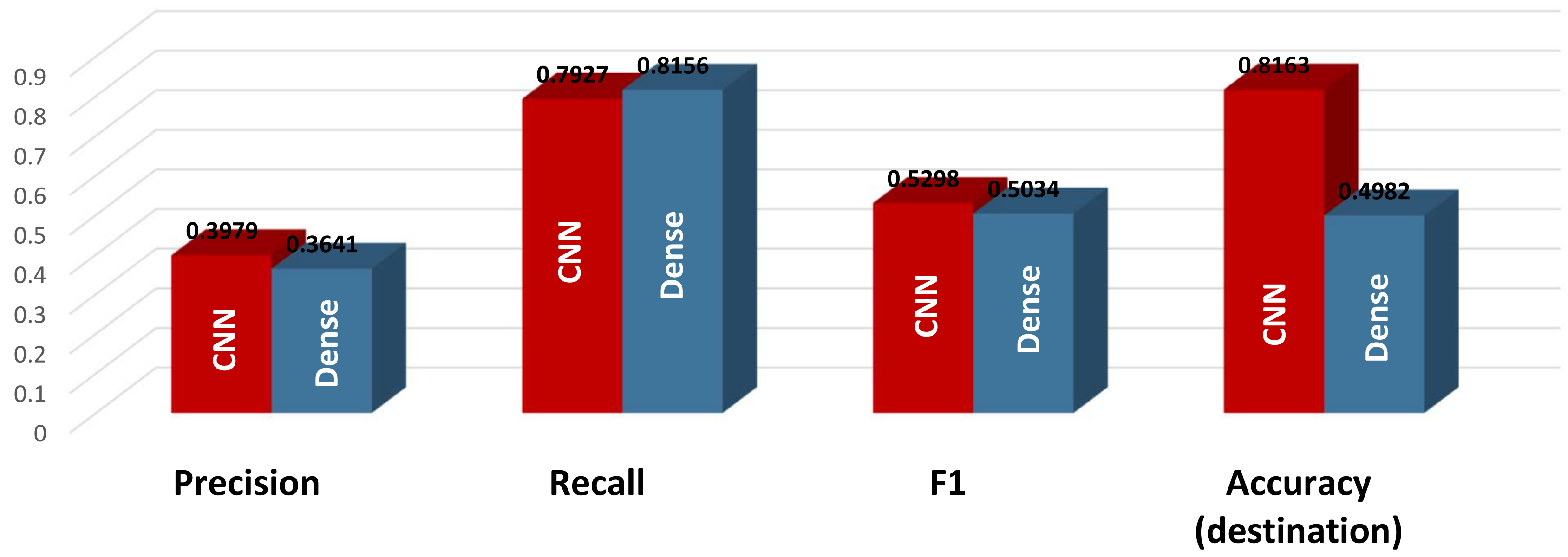


CNN VS Dense

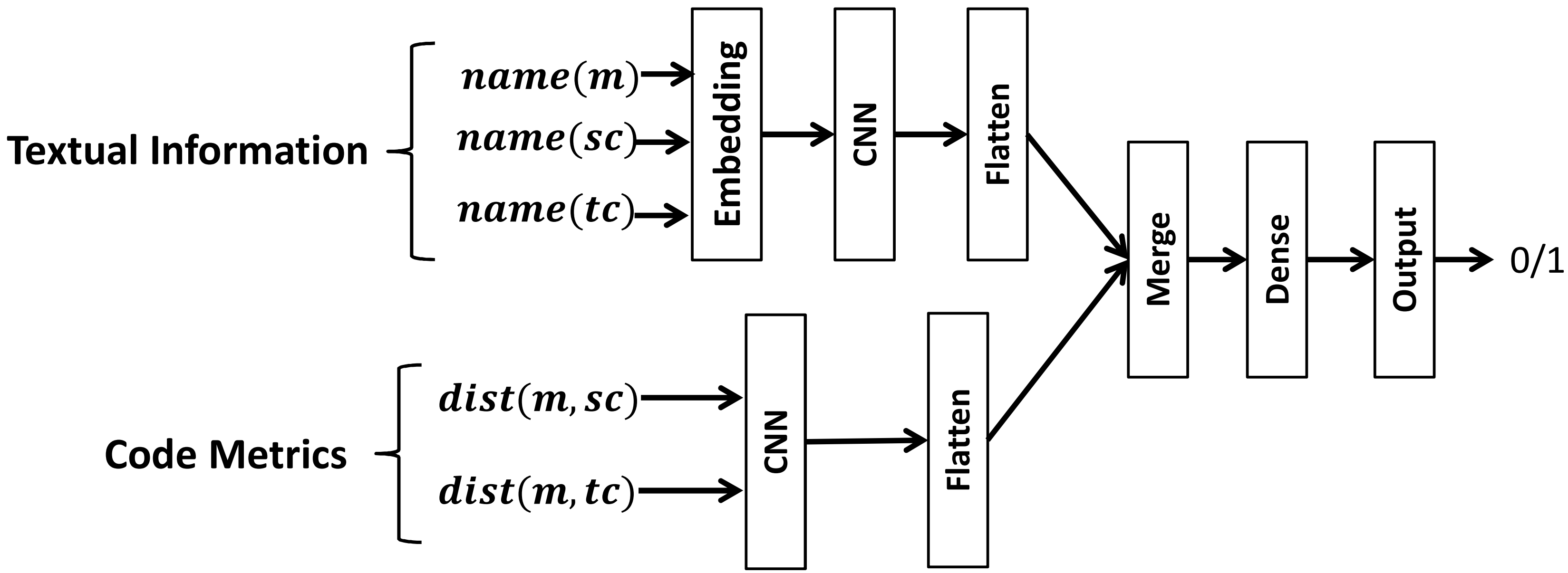


$name(m)$
$name(sc)$
$name(tc)$
Embedding
CNN
Flatten
$dist(m, sc)$
$dist(m, tc)$
Dense
Flatten
Merge
Dense
Output
0/1

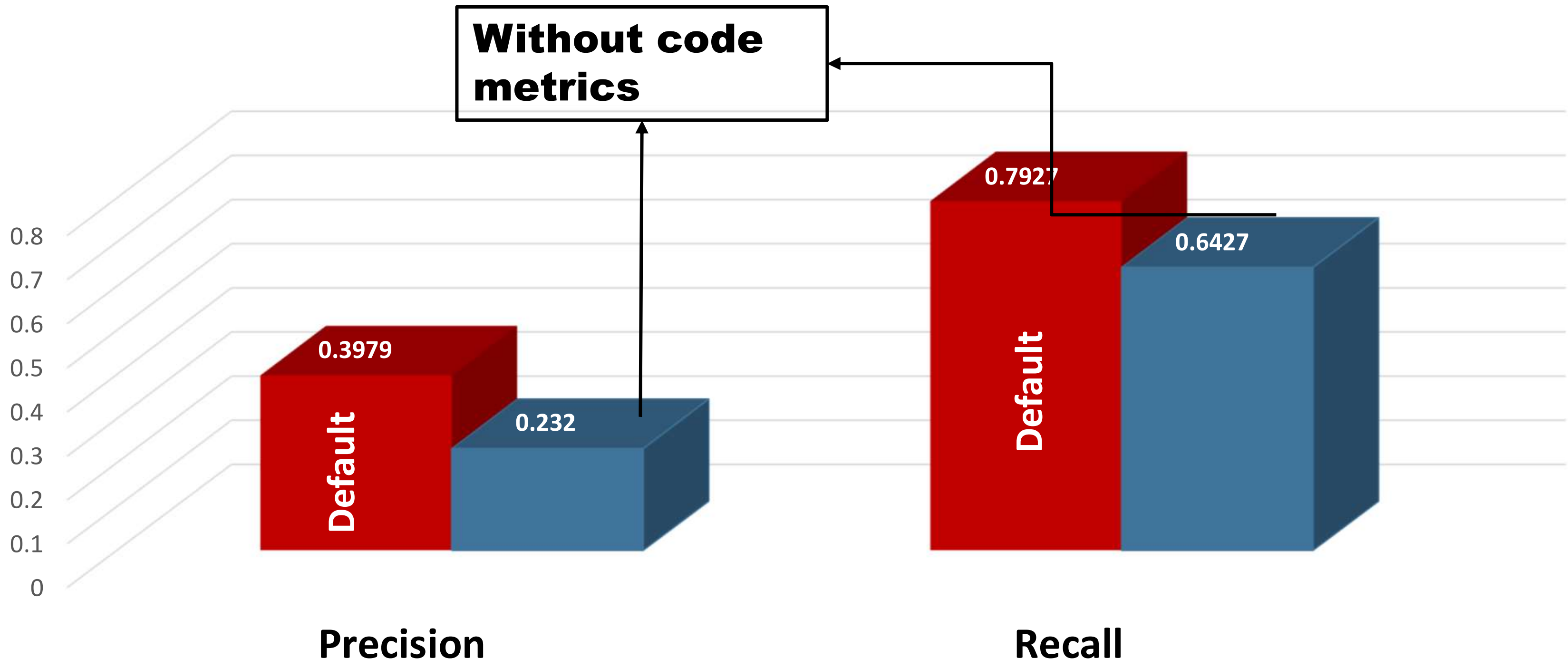
CNN VS Dense


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.3979
0.3641
CNN
Dense
0.7927
0.8156
CNN
Dense
0.5298
0.5034
CNN
Dense
0.8163
0.4982
CNN
Dense
Precision
Recall
F1
Accuracy (destination)

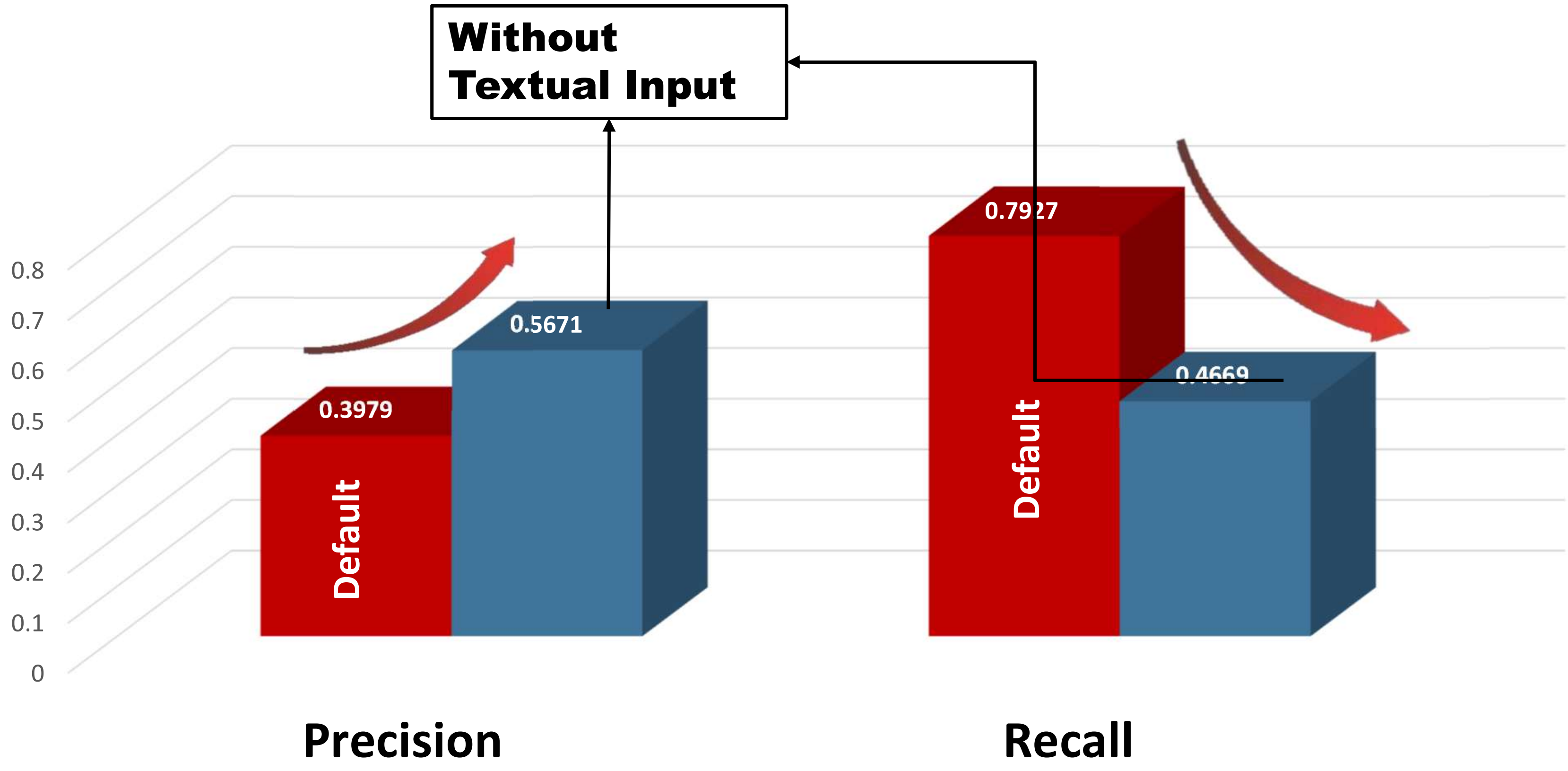
Influence of Code Metrics



Influence of Code Metrics



Influence of Textual Input



Case study

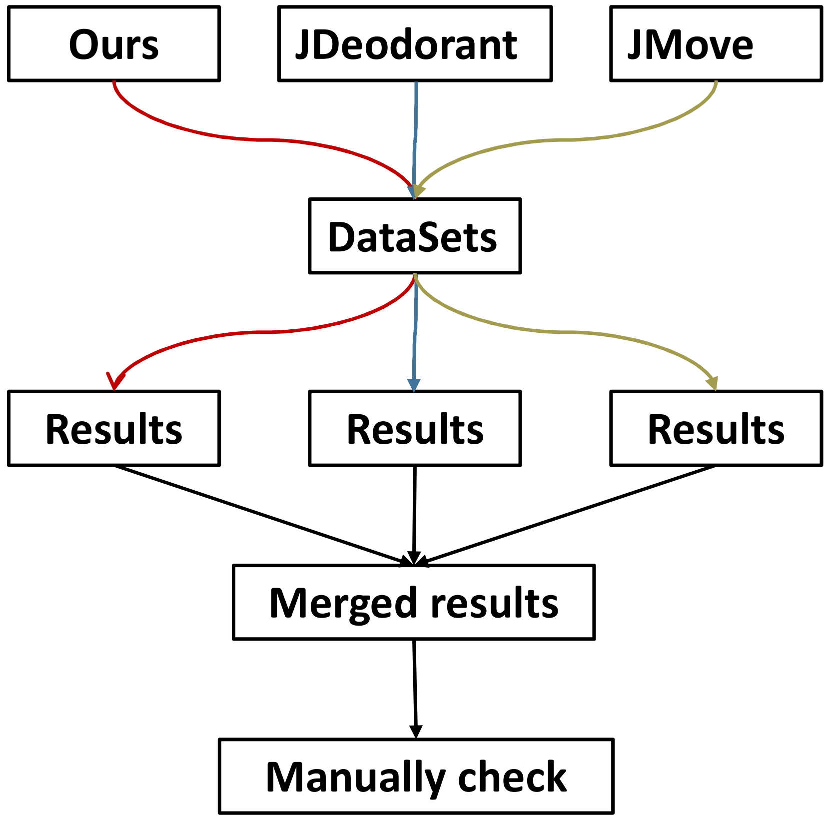
Generated data **VS** Real feature envy

Process

Applications	Domain	#Classes	#Methods	LOC
XDM	Download Manager	198	1599	42,604
JSmooth	Java Wrapper	91	669	16,782
Neuroph	Neural Network	214	1186	26,513

Process

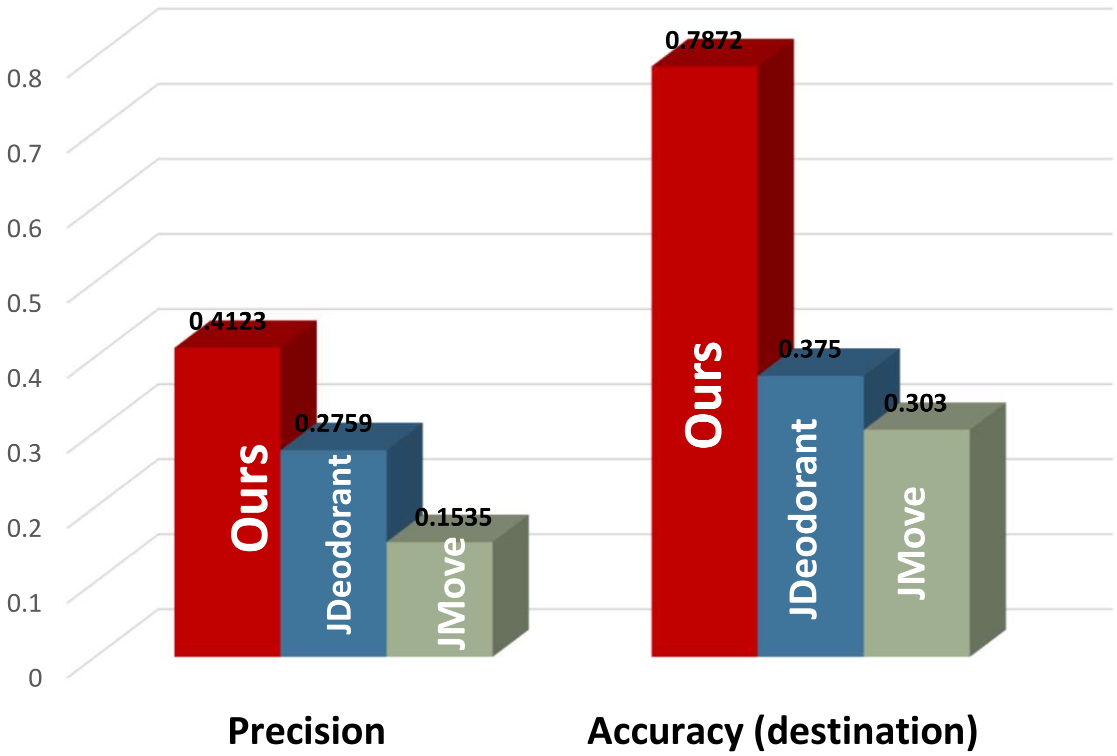
- Detect feature envy with our approach, JDeodorant and JMove independently
- Merge the detection results
- Manually check the detection results

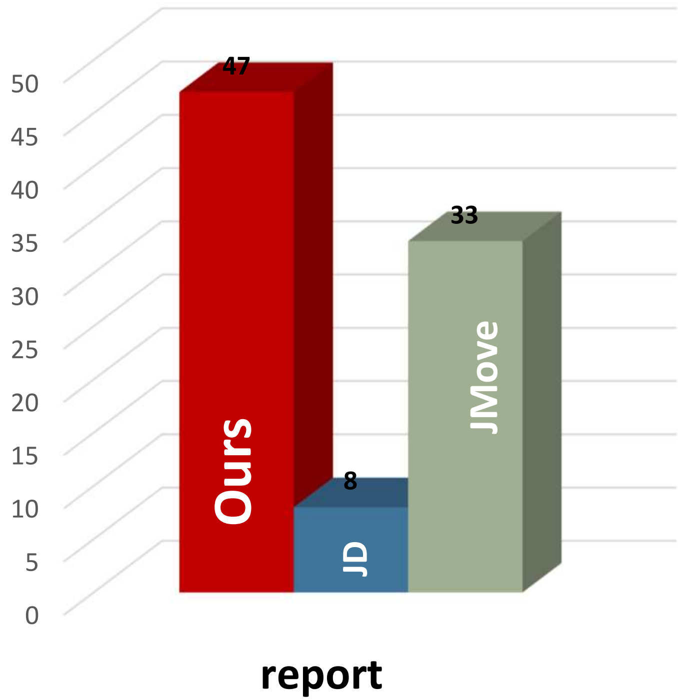


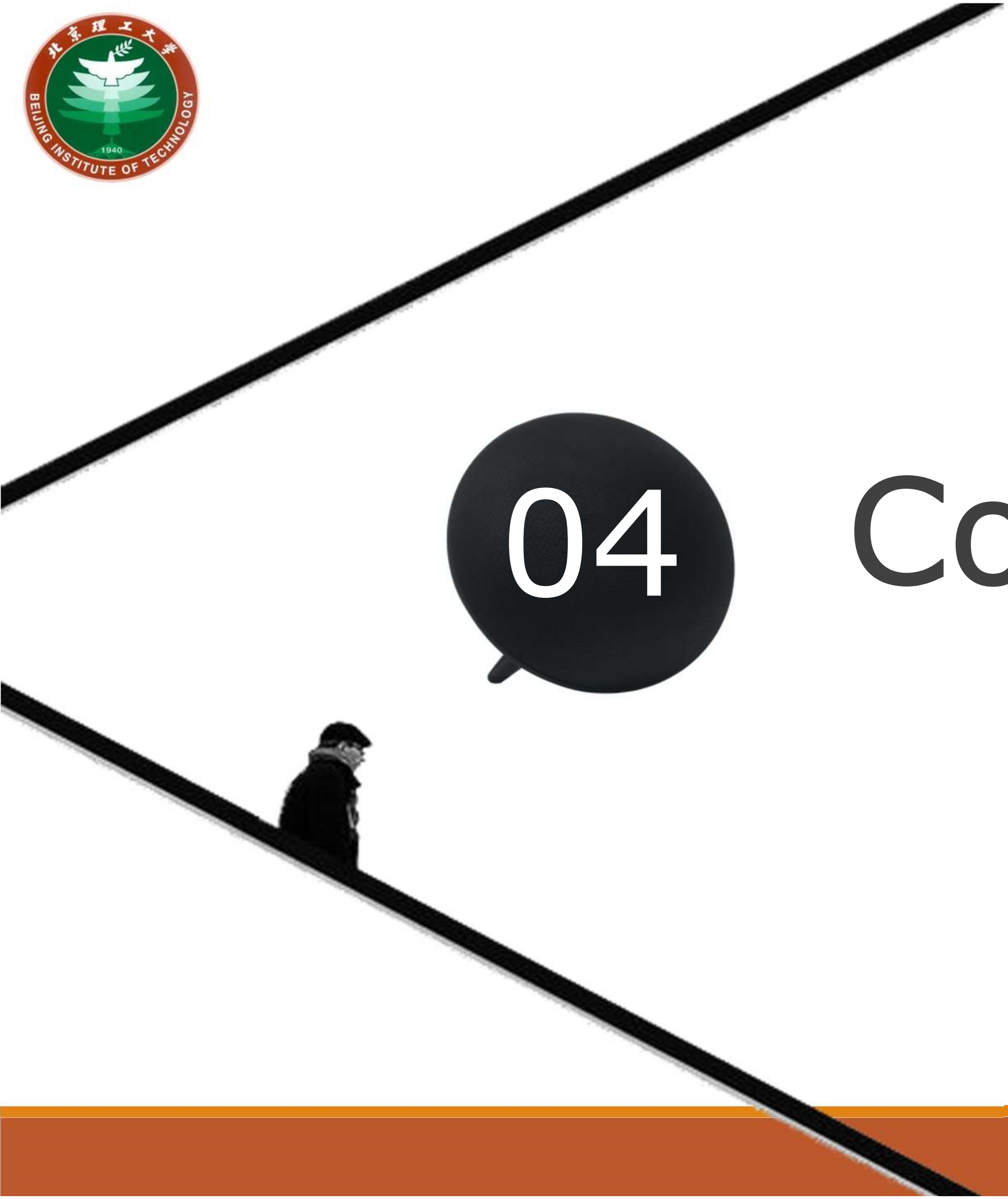
Results

Metrics	Ours	JDeodorant	JMove
#Reported	114	29	215
#Accepted	47	8	33
#Accepted targets	37	3	10
Precision	41.23%	27.59%	15.35%
Accuracy(destination)	78.27%	37.5%	30.30%

Results of Case Study







04 Conclusions

Take Away Messages

- ✓ **Deep learning improves the state of the art.**
- ✓ **Training data could be generate automatically.**

THANKS