Code Scent

A Customizable IntelliJ IDEA Plugin for Code Smell Detection

CSED 332: Software Design Methods
Project Final Presentation

4nix (Group 4): Jinyoung Kim, Chanho Song, Jinmin Goh, Hyunbin Park, Seokhwan Choi, Gwanho Kim

Table of contents

01

Introduction

Main goal of the project

Introduction

- **Problem 1: Code smells introduce bad design**
    - Code smells can cause problems when fixing bugs, adding new features, or making any changes to the code. Also, they might lead to new bugs.

- **Problem 2: Each user has different boundaries for what they consider a code smell**
    - For example, some people may think the appropriate number of parameters for a method is three or fewer, while others may think it is six or fewer

- **Problem 3: Users want to see code smells in their at a glance, categorized by type.**
    - In the traditional plugin, you can only see what a warning is by going to its line.

Introduction

- **Main goal of our project “Code Scent”**
    - Improve code quality by detecting code smells using automatic code smell detection plugin
    - Provide locations, explanations of what code smells are found
    - **Unique feature "customization"**: User can customize the precondition for certain code smells, so that the code smell condition fits to user's convenience (personalized custom plugin)
        - Can customize "large class due to fields", "large class due to methods", "long method", "message chain", "long parameter list", and "comments" code smells.

02

Overview of the Project

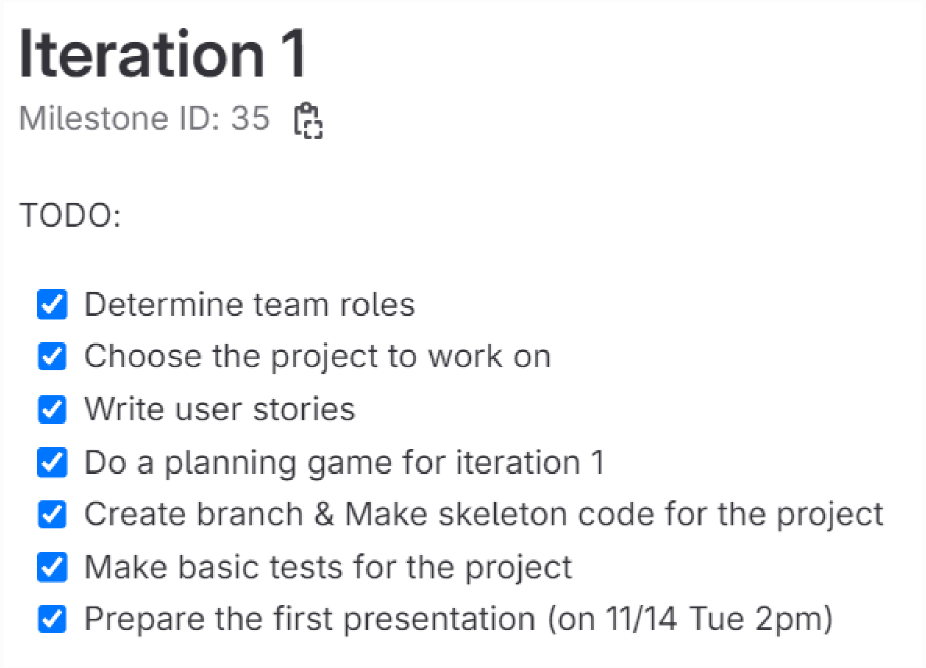
Distribution of user stories across iterations

Explain and evaluate XP practices applied by our team

Iteration 1

- Make the user story
- Find out how to make plugin project
- Set up skeleton structure for plugin

  project



Iteration 1

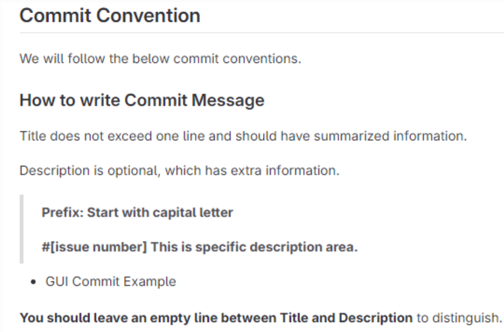
Milestone ID: 35

TODO:

- [x] Determine team roles
- [x] Choose the project to work on
- [x] Write user stories
- [x] Do a planning game for iteration 1
- [x] Create branch & Make skeleton code for the project
- [x] Make basic tests for the project
- [x] Prepare the first presentation (on 11/14 Tue 2pm)

XP Programming in Iteration 1

- Assigned priorities and estimated units with planning game by each pair (breakdown user stories into tasks)
- **Set proper conventions for developing (proper use of codelines and branching)**
    - Branch naming, commit, merge request, etc
- Held frequent meetings for iteration 1 period (6 meetings)
    - Gave continuous feedbacks for every meeting



Commit Convention

We will follow the below commit conventions.

How to write Commit Message

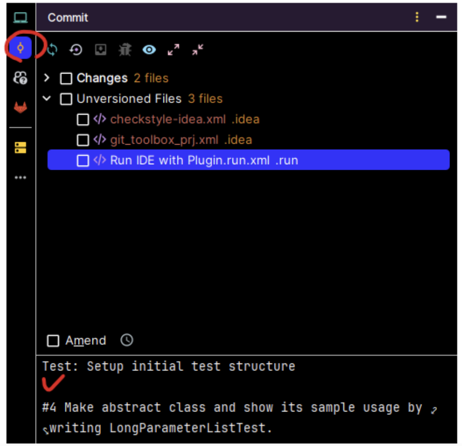
Title does not exceed one line and should have summarized information.

Description is optional, which has extra information.

> **Prefix: Start with capital letter**
>
> **#[issue number] This is specific description area.**

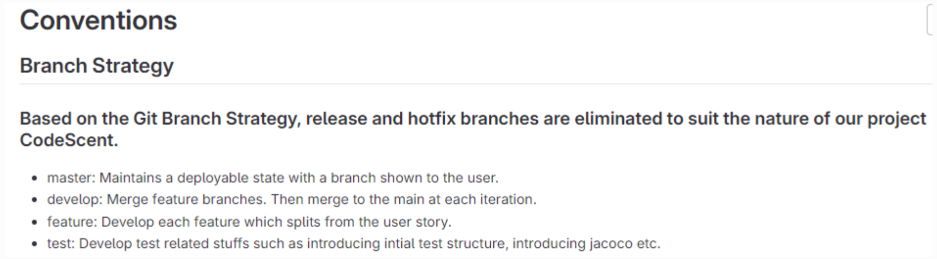
- GUI Commit Example

You should leave an empty line between Title and Description to distinguish.


Commit
Changes 2 files
Unversioned Files 3 files
checkstyle-idea.xml .idea
git_toolbox_prj.xml .idea
Run IDE with Plugin.run.xml .run
Amend

```
Test: Setup initial test structure

#4 Make abstract class and show its sample usage by
writing LongParameterListTest.
```

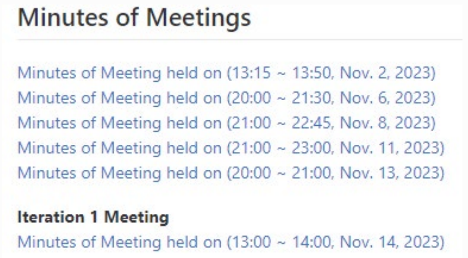


Conventions

Branch Strategy

Based on the Git Branch Strategy, release and hotfix branches are eliminated to suit the nature of our project CodeScent.

- master: Maintains a deployable state with a branch shown to the user.
- develop: Merge feature branches. Then merge to the main at each iteration.
- feature: Develop each feature which splits from the user story.
- test: Develop test related stuffs such as introducing intial test structure, introducing jacoco etc.



Minutes of Meetings

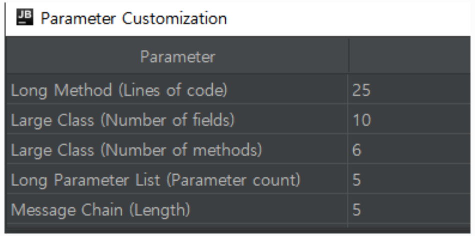
Minutes of Meeting held on (13:15 ~ 13:50, Nov. 2, 2023)
Minutes of Meeting held on (20:00 ~ 21:30, Nov. 6, 2023)
Minutes of Meeting held on (21:00 ~ 22:45, Nov. 8, 2023)
Minutes of Meeting held on (21:00 ~ 23:00, Nov. 11, 2023)
Minutes of Meeting held on (20:00 ~ 21:00, Nov. 13, 2023)

Iteration 1 Meeting

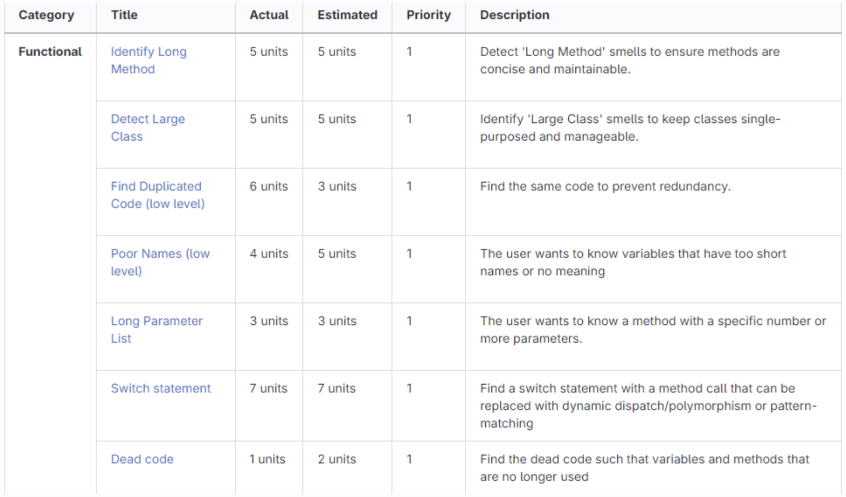
Minutes of Meeting held on (13:00 ~ 14:00, Nov. 14, 2023)

Iteration 2

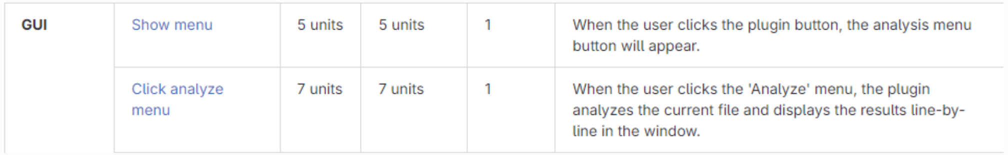
- Actual iteration 2 velocity = **51 units/iteration**, Estimated velocity = 52 units/iteration
- Usually, priority 1 user stories are implemented in iteration 2.
    - 7 user stories for code smell detection, 4 user stories for GUI
- 2 new user stories were added in iteration 2
    - Customization pop-up window, and checking user's input


Parameter Customization

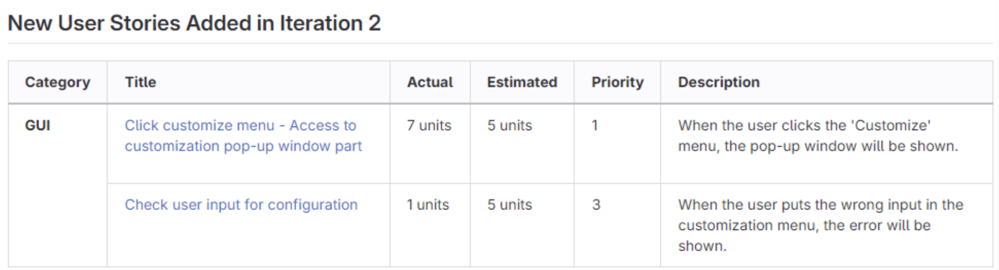
Parameter	
Long Method (Lines of code)	25
Large Class (Number of fields)	10
Large Class (Number of methods)	6
Long Parameter List (Parameter count)	5
Message Chain (Length)	5



Category	Title	Actual	Estimated	Priority	Description
Functional	Identify Long Method	5 units	5 units	1	Detect 'Long Method' smells to ensure methods are concise and maintainable.
	Detect Large Class	5 units	5 units	1	Identify 'Large Class' smells to keep classes single-purposed and manageable.
	Find Duplicated Code (low level)	6 units	3 units	1	Find the same code to prevent redundancy.
	Poor Names (low level)	4 units	5 units	1	The user wants to know variables that have too short names or no meaning
	Long Parameter List	3 units	3 units	1	The user wants to know a method with a specific number or more parameters.
	Switch statement	7 units	7 units	1	Find a switch statement with a method call that can be replaced with dynamic dispatch/polymorphism or pattern-matching
	Dead code	1 units	2 units	1	Find the dead code such that variables and methods that are no longer used



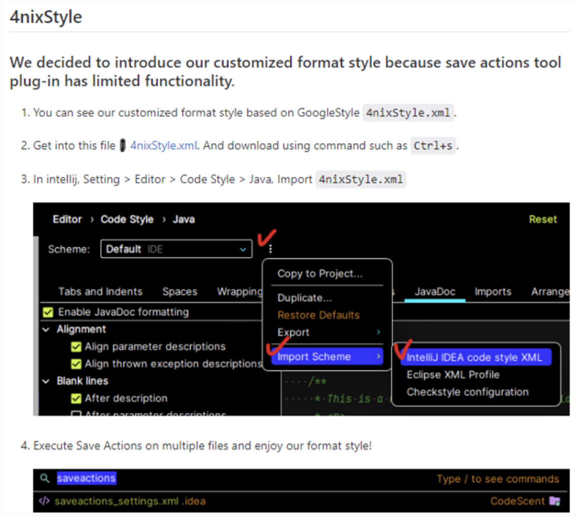
GUI	Show menu	5 units	5 units	1	When the user clicks the plugin button, the analysis menu button will appear.
	Click analyze menu	7 units	7 units	1	When the user clicks the 'Analyze' menu, the plugin analyzes the current file and displays the results line-by-line in the window.


New User Stories Added in Iteration 2

Category	Title	Actual	Estimated	Priority	Description
GUI	Click customize menu - Access to customization pop-up window part	7 units	5 units	1	When the user clicks the 'Customize' menu, the pop-up window will be shown.
	Check user input for configuration	1 units	5 units	3	When the user puts the wrong input in the customization menu, the error will be shown.

XP Programming in Iteration 2

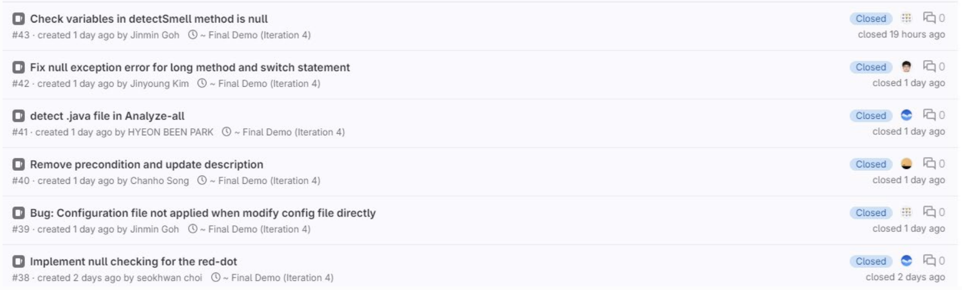
- Pair programming
  - Worked in each pair for dedicated user stories (breakdown user stories into tasks)
  - Each pair meets regularly to review the code they are developing and merge requests approved between pairs
- **Code formatter (coding standards)**
  - Used “Save Actions Tool” plug-in for maintaining consistency of code format structure
- Test driven development
  - Setup initial test structure first (test fails), write code, test pass, refactor
- Continuous feedback and communication, continuous integration
  - Held 5 meetings for iteration 2, merge frequently
- Refactor duplicated utilities to utils directory (LoadPsi.java)
- **Make “issues” when developing for tracking milestone plans**



4nixStyle

We decided to introduce our customized format style because save actions tool plug-in has limited functionality.

1. You can see our customized format style based on GoogleStyle `4nixStyle.xml`.
2. Get into this file 4nixStyle.xml. And download using command such as `Ctrl+s`.
3. In intellij, Setting > Editor > Code Style > Java, Import `4nixStyle.xml`
4. Execute Save Actions on multiple files and enjoy our format style!



Check variables in detectSmell method is null
#43 · created 1 day ago by Jinmin Goh ~ Final Demo (Iteration 4)
Closed 0
closed 19 hours ago

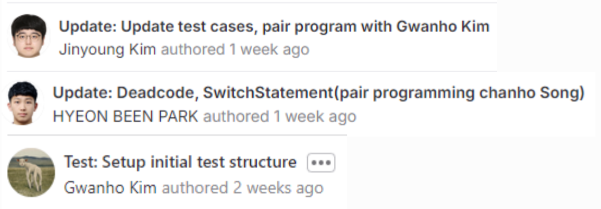
Fix null exception error for long method and switch statement
#42 · created 1 day ago by Jinyoung Kim ~ Final Demo (Iteration 4)
Closed 0
closed 1 day ago

detect .java file in Analyze-all
#41 · created 1 day ago by HYEON BEEN PARK ~ Final Demo (Iteration 4)
Closed 0
closed 1 day ago

Remove precondition and update description
#40 · created 1 day ago by Chanho Song ~ Final Demo (Iteration 4)
Closed 0
closed 1 day ago

Bug: Configuration file not applied when modify config file directly
#39 · created 1 day ago by Jinmin Goh ~ Final Demo (Iteration 4)
Closed 0
closed 1 day ago

Implement null checking for the red-dot
#38 · created 2 days ago by seokhwan choi ~ Final Demo (Iteration 4)
Closed 0
closed 2 days ago



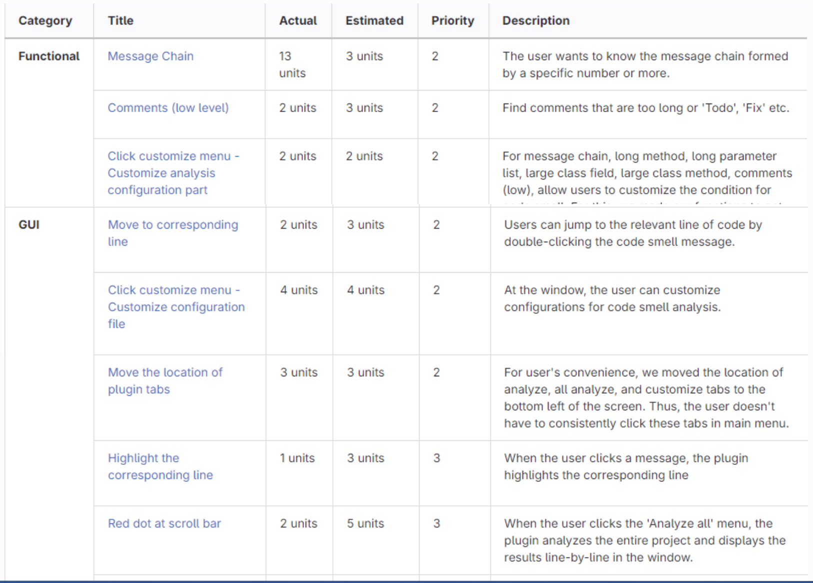
Update: Update test cases, pair program with Gwanho Kim
Jinyoung Kim authored 1 week ago

Update: Deadcode, SwitchStatement(pair programming chanho Song)
HYEON BEEN PARK authored 1 week ago

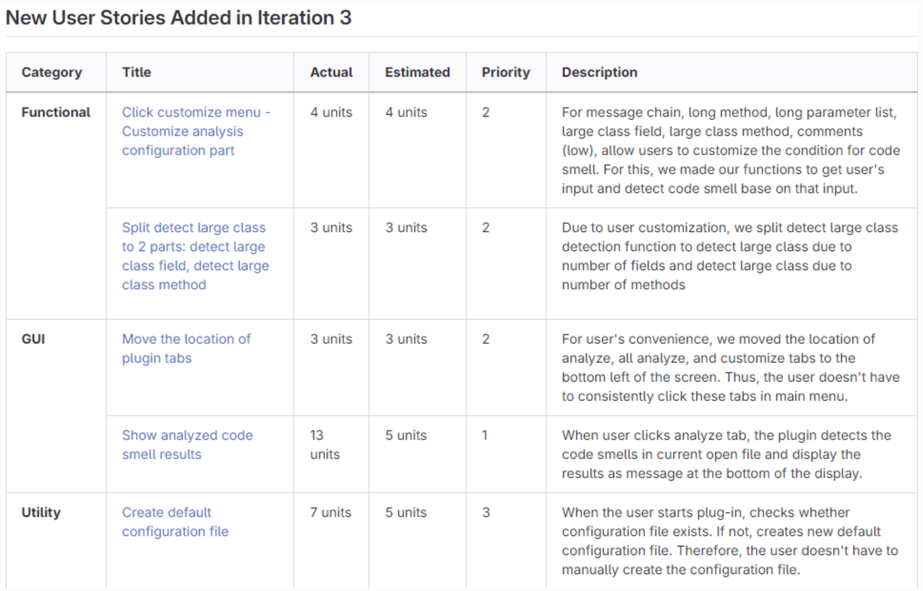
Test: Setup initial test structure
Gwanho Kim authored 2 weeks ago

Iteration 3

- Actual iteration 2 velocity = **53 units/iteration**, Estimated velocity = 43 units/iteration
- Usually, priority 2 user stories are implemented in iteration 3.
    - 4 user stories for code smell detection, 7 user stories for GUI, 1 user story for utility
- 5 new user stories were added in iteration 3



Category	Title	Actual	Estimated	Priority	Description
Functional	Message Chain	13 units	3 units	2	The user wants to know the message chain formed by a specific number or more.
	Comments (low level)	2 units	3 units	2	Find comments that are too long or 'Todo', 'Fix' etc.
	Click customize menu - Customize analysis configuration part	2 units	2 units	2	For message chain, long method, long parameter list, large class field, large class method, comments (low), allow users to customize the condition for
GUI	Move to corresponding line	2 units	3 units	2	Users can jump to the relevant line of code by double-clicking the code smell message.
	Click customize menu - Customize configuration file	4 units	4 units	2	At the window, the user can customize configurations for code smell analysis.
	Move the location of plugin tabs	3 units	3 units	2	For user's convenience, we moved the location of analyze, all analyze, and customize tabs to the bottom left of the screen. Thus, the user doesn't have to consistently click these tabs in main menu.
	Highlight the corresponding line	1 units	3 units	3	When the user clicks a message, the plugin highlights the corresponding line
	Red dot at scroll bar	2 units	5 units	3	When the user clicks the 'Analyze all' menu, the plugin analyzes the entire project and displays the results line-by-line in the window.

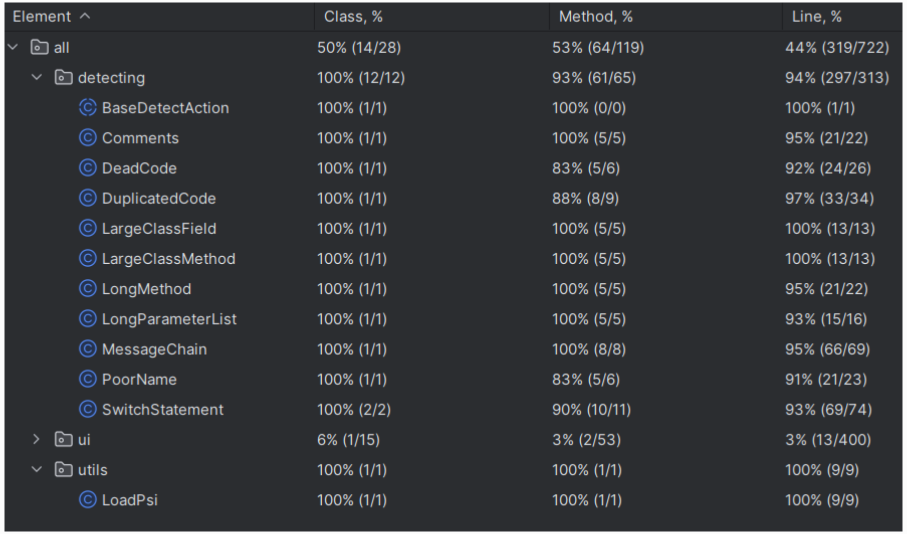


New User Stories Added in Iteration 3

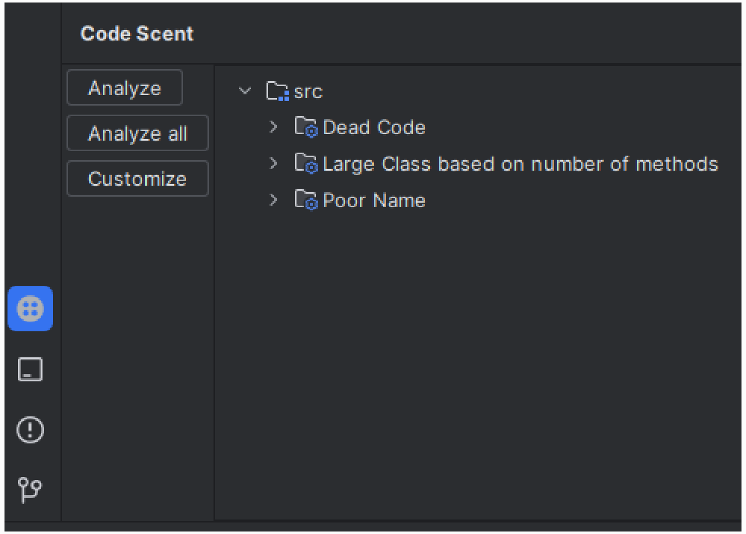
Category	Title	Actual	Estimated	Priority	Description
Functional	Click customize menu - Customize analysis configuration part	4 units	4 units	2	For message chain, long method, long parameter list, large class field, large class method, comments (low), allow users to customize the condition for code smell. For this, we made our functions to get user's input and detect code smell base on that input.
	Split detect large class to 2 parts: detect large class field, detect large class method	3 units	3 units	2	Due to user customization, we split detect large class detection function to detect large class due to number of fields and detect large class due to number of methods
GUI	Move the location of plugin tabs	3 units	3 units	2	For user's convenience, we moved the location of analyze, all analyze, and customize tabs to the bottom left of the screen. Thus, the user doesn't have to consistently click these tabs in main menu.
	Show analyzed code smell results	13 units	5 units	1	When user clicks analyze tab, the plugin detects the code smells in current open file and display the results as message at the bottom of the display.
Utility	Create default configuration file	7 units	5 units	3	When the user starts plug-in, checks whether configuration file exists. If not, creates new default configuration file. Therefore, the user doesn't have to manually create the configuration file.

Iteration 3

- Improve automatic test coverage by **adding more test data**
- Implement new function (Message chain, comments, duplicated code, switch statement)
- Improve UI (tree structure based on the code smell type)

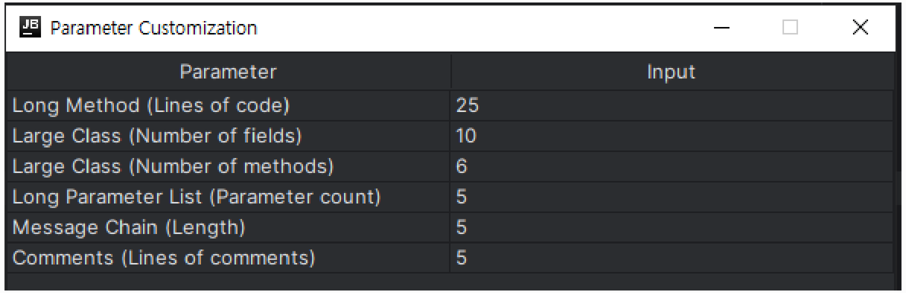


Element	Class, %	Method, %	Line, %
all	50% (14/28)	53% (64/119)	44% (319/722)
detecting	100% (12/12)	93% (61/65)	94% (297/313)
BaseDetectAction	100% (1/1)	100% (0/0)	100% (1/1)
Comments	100% (1/1)	100% (5/5)	95% (21/22)
DeadCode	100% (1/1)	83% (5/6)	92% (24/26)
DuplicatedCode	100% (1/1)	88% (8/9)	97% (33/34)
LargeClassField	100% (1/1)	100% (5/5)	100% (13/13)
LargeClassMethod	100% (1/1)	100% (5/5)	100% (13/13)
LongMethod	100% (1/1)	100% (5/5)	95% (21/22)
LongParameterList	100% (1/1)	100% (5/5)	93% (15/16)
MessageChain	100% (1/1)	100% (8/8)	95% (66/69)
PoorName	100% (1/1)	83% (5/6)	91% (21/23)
SwitchStatement	100% (2/2)	90% (10/11)	93% (69/74)
ui	6% (1/15)	3% (2/53)	3% (13/400)
utils	100% (1/1)	100% (1/1)	100% (9/9)
LoadPsi	100% (1/1)	100% (1/1)	100% (9/9)



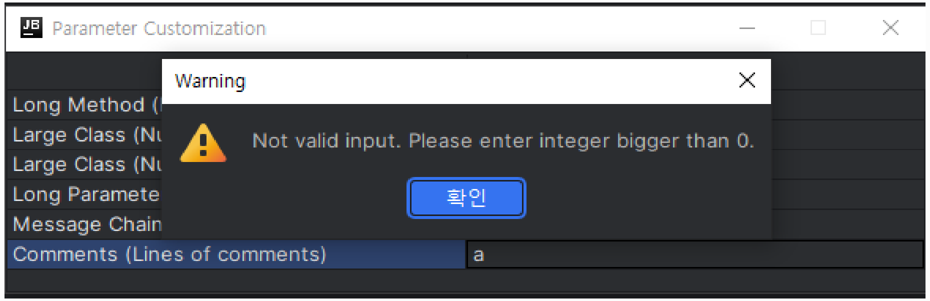
Iteration 3

- **Implementation of user customization part**
  - **Receives user input** and detects code smells based on it
  - It works based on the properties file
  - Implementation of exception handling for incorrect input



Parameter Customization

Parameter	Input
Long Method (Lines of code)	25
Large Class (Number of fields)	10
Large Class (Number of methods)	6
Long Parameter List (Parameter count)	5
Message Chain (Length)	5
Comments (Lines of comments)	5



Parameter Customization

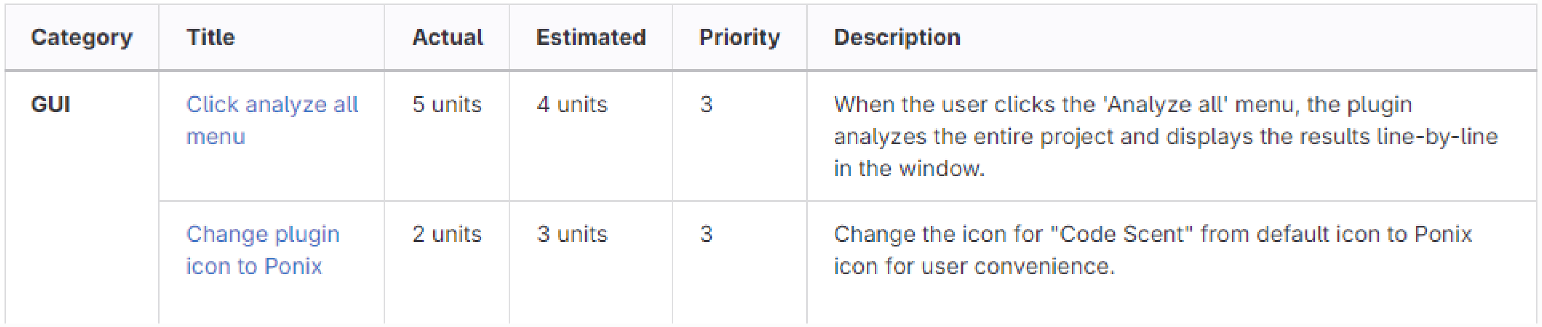
Warning

Not valid input. Please enter integer bigger than 0.

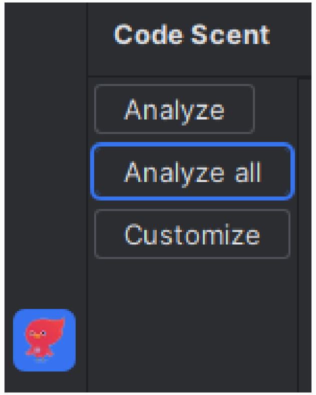
Long Method (	
Large Class (Nu	
Large Class (Nu	
Long Paramete	
Message Chain	
Comments (Lines of comments)	a

Iteration 4

- Actual iteration 4 velocity = **7 units/iteration**, Estimated velocity = 7 units/iteration
- Priority 3 user story is implemented in iteration 4.
  - 1 new user story is added
- More focused on **refactoring and quality of the project** (Finding and fixing bugs, write clean comments)

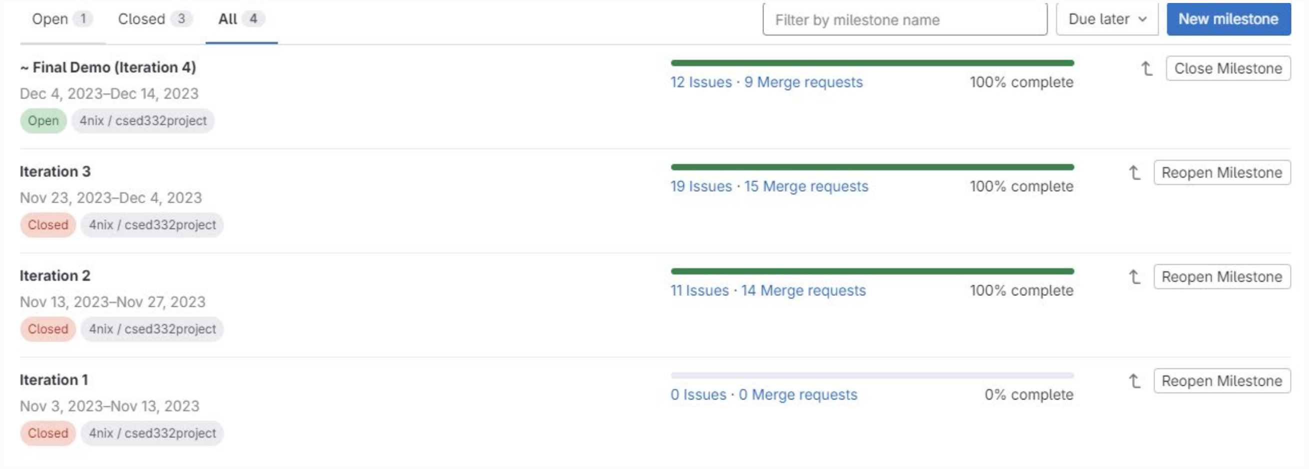


Category	Title	Actual	Estimated	Priority	Description
GUI	Click analyze all menu	5 units	4 units	3	When the user clicks the 'Analyze all' menu, the plugin analyzes the entire project and displays the results line-by-line in the window.
	Change plugin icon to Ponix	2 units	3 units	3	Change the icon for "Code Scent" from default icon to Ponix icon for user convenience.



Result

- Total number of user stories developed in iteration 1: 24 user stories
- Total number of user stories actually implemented: 25 user stories (9 new user stories added)
  - Total units: 111 units
- Total number of user stories not implemented from iteration 1: 8 user stories for functionality



~ Final Demo (Iteration 4)
Dec 4, 2023–Dec 14, 2023
Open · 4nix / csed332project
12 Issues · 9 Merge requests · 100% complete
Close Milestone

Iteration 3
Nov 23, 2023–Dec 4, 2023
Closed · 4nix / csed332project
19 Issues · 15 Merge requests · 100% complete
Reopen Milestone

Iteration 2
Nov 13, 2023–Nov 27, 2023
Closed · 4nix / csed332project
11 Issues · 14 Merge requests · 100% complete
Reopen Milestone

Iteration 1
Nov 3, 2023–Nov 13, 2023
Closed · 4nix / csed332project
0 Issues · 0 Merge requests · 0% complete
Reopen Milestone

03

Project Demo

Demonstrating our “Code Scent”