Opening various image formats

Software Maintance Autumn 2023

Magnus Christian Larsen

January 1, 2024

GitHub Username: Autowinto
Student Mail: magla21@student.sdu.dk
Project: https://github.com/Autowinto/JHotDraw
Date: mm/dd/yyyy

Contents

1 Change Request

For this project in the Software Maintanence course, I decided to refactor the feature that allows the user to open and load various files into the software. The feature itself had several subfeatures, namely subfeatures for jpg, png, svg, text etc., but for this exact report, I decided to focus specifically on PNG files.

1.1 User Story

The change request, defined as a user story, is as follows:

"As a user, I want to be able to open images of the formats png, jpg, gif, pct, and text, so that I can work with many different formats without having to convert myself"

1.2 Acceptance Criteria

The following subtasks as identified serve as acceptance criteria:

- When I click on the 'Open' option, the software should allow me to browse and select files in png, jpg, gif, pct, and text formats.
- Upon selecting a valid file of the mentioned formats, the software should display the image or content correctly within the workspace.
- If I attempt to open a corrupted or unsupported file type, the software should provide a clear error message.
- The software should maintain the original quality of the image when opening it.
- For text files, the software should provide an option to convert the text into an editable shape or object in the drawing workspace.

2 Concept Location

2.1 General Methodology

In locating classed for the refactoring process of the report, there are several options available to us within Featureous and NetBeans, which we of course will make use of in the future impact analysis section of the report. For actually finding the location of classes relevant to the feature, I mainly made use of the powerful tools provided by the IntelliJ IDE, such as the following:

- Find usages - Allows you to find usages of any given method or variable
- Go to implementation - Allows you to go directly to the implementation of a method
- Search within file - Allows you to search for any symbol or token within a single file.
- Inspect code - Allows you to see a bunch of metrics about code such as maturity, implementation issues and more, helping to pinpoint problematic parts of the code.

In addition to tooling provided by NetBeans and IntelliJ, there are also methods that are much less technical, such as simply removing parts of the code and seeing what happens and if said code is relevant to the feature that is being refactored.

2.2 Classes Overview

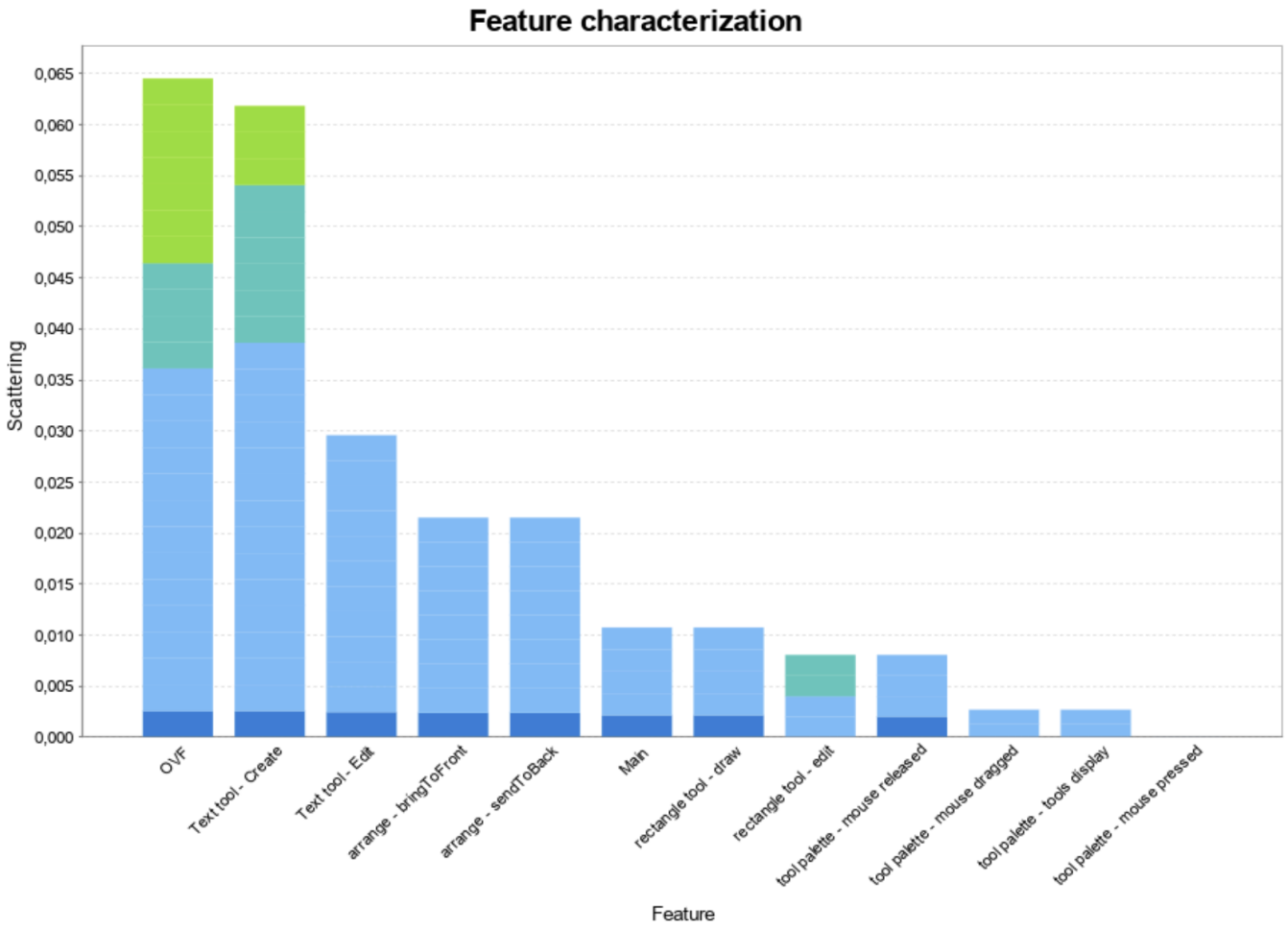
#	**Domain classes**	**Tool used**	**Comments**
1	LoadFileAction		
2	OpenFileAction		
3	AbstractApplicationAction		
4	AbstractAction		

Use Featureous Feature call-tree to provide a tree-based visualization of the runtime call graphs of methods implementing the features of your change request. This view provides an execution-based alternative to the hierarchical fashion of browsing features supported by the mentioned feature inspector view.

3 Impact Analysis

The following impact analysis serves the purpose of analzying stuff TODO: Write more DESCRIBE THE FEATURE ENDPOINTS USED

3.1 Featureous Feature-Code Characterization



Use Featureous Feature-Code Characterization View to illustrate Scattering and Tangling of each feature that is involved in your Change Request. Write your reflections about how these measurements relate to your feature and future refactorings. For example, text about the purpose of Impact Analysis and ripple effects.

Use Featureous Feature Relations Characterization view to relate your change request features to each other with respect to how they depend on the objects created by other features and how they exchange data by sharing these objects with one another.

3.2 Feature Correlation Grid

3.3 Feature Correlation Graph

Use Feature-code correlation graph and feature-code correlation grid to illustrate detailed investigations of the correspondences between source code units and related features to your change request.

Using table 2 list the packages and their number of classes that you visited after you located the concept. Write short comments explaining what you have learned about each package and how they contribute to your feature?

3.4 Feature Relations Characterization

3.5 Dependency Analysis

Show the features that my feature depends on

3.6 Impact Analysis

In the fourth column, mention if the class is related to the concept. Use one of the following terms:

- Use "Unchanged" if the class has no relation to the concept but you have visited it.
- Use "Propagating" if you read the source code of the class and it guides you to the location of the concept, but you will not change it.
- Use "Changed" if the class will be changed.

Table 1: The list of all the packages visited during impact analysis.

Package name	**# of classes**	**Tool used**	**Comments**

4 Refactoring Patterns and Code smells

There is so much fucking duplicated code. The methods are incredibly long Method names are fucked and have no proper meaning Long parameter lists Confusing naming of classes and functions.

There is no reason to be abbreviating variables to the point of them being single letters. It makes for confusing code, especially in longer methods.

Describe the code smell that triggered your refactoring, see Chapter 4 in [Ker05]. Describe what you plan to change by refactoring. Why do you think it's good to do the refactoring? Describe the strategy of the refactorings. Remember there is not one way to implement a pattern. Which of the refactorings from [Ker05] did you apply and what was the reasoning behind it?

5 Refactoring Implementation

Explain where and why you made changes in the source code. That is, explain the difference between the actual change set compared to the estimated impacted set from the impact analysis. Which classes or methods did you create or change? Furthermore, describe used design patterns and reflect about improved design to improve future software maintenance.

6 Verification

At class level document unit tests of important business functionality. Document how you have verified your implemented change. Document the results of your acceptance test that test your feature from your change request.

7 Continuous Integration

Reflect on the following questions:

- What is Continuous Integration and how could you apply it?
- How to use version control systems (git) based on your portfolio?

8 Conclusion

Explain your experience with creating the final baseline of JHotDraw. How did you manage to merge in your changes to the baseline ? How was the system tested? Describe your reflection on what went well and what went not so well. Did you have to cut the scope of your change request and did you have to put some issues back into the backlog. What can be done to avoid future problems, i.e. what have you learned from the iteration.

9 Source Code

Source Repository: https://github.com/Autowinto/JHotDraw **Feature Branch:** https://github.com/Auto
magnus