Change request log

# Team

Jacob, Travis

# Change Request

jEdit CR 2 – allow text matching in the Recent Files menu to match the input string against sections of filenames other than the start

# Concept Location

Use the table below to describe each step you follow when performing concept location for this change request. In your description, include the following information when appropriate:

* IDE Features used (e.g., searching tool, dependency navigator, debugging, etc.)
* Queries used when searching
* System executions and input to the system
* Interactions with the system (e.g., pages visited)
* Classes visited
* The first class found to be changed (this is when concept location ends)

When there is a major decision/step in the process, include its rationale, i.e., why that decision/step was taken.

Make sure you time yourselves when going through this process and provide the total time spent below.

The following is an example of a concept location process for the change request "Color student schedule":

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We ran the program | This helped us develop an understanding of the program’s current behavior |
| 2 | We searched the source code for terms such as “Recent Files” | This was what we thought we allow us to find the relevant source where pattern matching was performed |
| 3 | We searched for “start” in RecentFilesProvider.java | This would allow us to see if the code explicitly required a string to match the start of the line |
| 4 | Relevant code found |  |

**Time spent (in minutes):** 30

# Impact Analysis

Use the table below to describe each step you follow when performing impact analysis for this change request. Include as many details as possible, including why classes are visited or why they are discarded from the estimated impact set.

Do not take the impact analysis of your changes lightly. Remember that any small change in the code could lead to large changes in the behavior of the system. Follow the impact analysis process covered in the class. Describe in details how you followed this process in the change request log. Provide details on how and why you finished the impact analysis process.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We searched for uses of the RecentFilesProvider in the codebase | We needed to know what other code may need to be changed to prevent unexpected behavior |
| 2 | None other than the desired menu panel used the provider | No further impact analysis is necessary |

**Time spent (in minutes):** 5

# Actualization

Use the table below to describe each step you followed when changing the code. Include as many details as possible, including why classes/methods were modified, added, removed, renamed, etc.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Wildcards were added to both ends of the regex to ensure it could match any position of the string | The current behavior only matched at the start, when wildcards at both ends would match both |

**Time spent (in minutes):** 15

# Validation

Use the table below to describe any validation activity (e.g., testing, code inspections, etc.) you performed for this change request. Include the description of each test case, the result (pass/fail) and its rationale.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We opened several files with similarities in their names | This is setup for the actual testing |
| 2 | Searching that should match at start | Expected behavior |
| 3 | Searching that should match at the middle | Expected behavior |
| 4 | Searching that should match nothing | Expected behavior |

**Time spent (in minutes):** 10

# Timing

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 30 |
| Impact Analysis | 5 |
| Actualization | 15 |
| Verification | 10 |
| Total | 60 |

# Reverse engineering

Create a UML sequence diagram (or more if needed) corresponding to the main object interactions affected by your change.

Create a partial UML class diagram of the classes visited while navigating through the code. Include the associations between classes (e.g., inheritance, aggregations, compositions, etc.), as well as the important fields and methods of each class that you learn about. The diagram may have disconnected components. Use the UML tool of your preference. When a significant fact about a class or method is learned, indicate it via annotations on the diagram. **For each change request, start with the diagram produced in the previous change request. For the first, you will start from scratch.**

Diagram

Description automatically generated

# Conclusions

Perform and analysis of the change requests and the change process. List the major challenges this change request posed.

List all the classes and methods you have changed.

The bulk of the work was performed before anything listed in the document was performed, namely, importing 5.5.0 to GitHub, attempting and failing to get it to work, restarting this process with 5.6.0, and getting that to compile and run. After this, locating the relevant code and performing the changes was fairly easy. Concept location was the hardest, as it took some time to familiarize ourselves with the project.

Classes and methods changed:

* org/gjt/sp/jedit/menu/RecentFilesProvider.java/RecentFiles/Provider
  + void update(menu:JMenu)