Change request log

# Team

Pilot – Dax, Navigator - Diego

# Change Request

Change request #1, change the status bar to include

# 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 used File Explore to search for the phrase “statusbar”* | *Trying to see if the status bar has its own dedicated script* |
| 2 | *We searched though each of the files under statusbar folder* | *perhaps one of the “widget” scripts is the line counter* |
| 3 | *We searched for the “(” character in statusbar.java* | *The status bar shows a line count with parentheses so maybe we can find were those are written to find what we need* |
| 4 | *replaced the “()” with “[]”, built, run then undo change* | *Need to verify that the parentheses are the ones we are looking for* |

**Time spent (in minutes):** 1 hour

# 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 tried to find out how to get the text then traced the code to see if would change anything* | *To track the classes that could be impacted by the change.* |
| 2 | *We saw that the getText() function does nothing else but return the string of the text area* | *We need the whole info to do a proper count since there was a word count option* |
| 3 | *In jEditTextArea.doWordCount() it counts the words but doesn’t return the results and creates a window to show results.* | *We didn’t want to change any of jEditTextArea code since it seemed important but would create complications* |

**Time spent (in minutes):** 40 min

# 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 | *Created a char array using the textArea.getText().toCharArray()* | *We needed to get the text area as a char array to count the words* |
| 2 | *Created two ints to count the words* |  |
| 3 | *Created a for loop that goes through the char array and whenever it finds a letter after a white space, increase the counters* | *This is how it counts words* |
| 4 | *Printed the results using buffer.append() just like how it prints the line counter* | *The buffer is what prints to the status bar so we just used their way of doing it* |

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

# 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 | *Inputs: “one two |three”*  *Expected output: (1/3)* | *Basic test*  *Test passed* |
| 2 | *Inputs: “one | two three”*  *Expected output: (1/3)* | *What happens in white space test passed* |
| 3 | *Inputs: “a |a a”*  *Expected output: (2/3)* | *Single words test passed* |
| 4 | *Inputs: “ |”*  *Expected output: (0/0)* | *No words Test passed* |

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

# Timing

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 60 min |
| Impact Analysis | 40 min |
| Actualization | 10 min |
| Verification | 10 min |
| Total | 120 min |

# 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.

*While changing this, we learned a bit more about the file structure of the application. We were able to find the concept location relatively easily by searching for the status bar. We used IntelliJ IDEA and ANT to modify and build the code, and we did some manual testing of the code to determine whether it had been working or not.*

*Classes and methods changed:*

* *org/gjt/sp/jedit/gui/StatusBar.java/StatusBar*
  + *void UpdateCaretStatus ()*