Change request Log

1. Concept Location

Step #	Description	Rationale
1	After installation, we ran the code to identify and understand what changes were to be made.	
2	After building the next thing we tried was to check if we could debug the code.	If we could attach a debugger we could get to know how the code flows.
3	Looking at the change required and the ui we did not find any clues on how to identify the source code responsible for this.	If we got any clue it would be easy for us in concept localization.
4	After carefully inspecting the jedit application we came across a feature that was similar tour change request.	In the gui there is a dialog box that displays the word count, character count and line count.
4	We started looking for code responsible for this dialog box.	Our assumption was there would be a common function using which the dialog box and the caret count are displaying the data.
5	We then looked into the actions.xml file to map how the dialog box is rendering information.	We found the function showWordCountDialog in jEditTextArea was responsible for word count dialog box.
6	Looking into the method we realized that the logic to count words and lines was here within the doWordCount function.	We then searched for all files to find where else this doWordCount function was being used. To our disappointment, there was nowhere else this function was being used.
7	We then followed a different approach and started searching for words such as caret throughout the code.	From the search result, we came across the getCaretPosition function.
8	We started looking for places where getCaretPosition was being invoked in the source code.	After understanding the functionality of this function we were sure that the change we had to make would use this function.
9	The search was narrowed down to the code files in GUI folder.	We initially searched for all the files but then realized that the change in question is a GUI and the narrowed our search to the gui folder.
10	We found the file statusBar.java using the function getCaretPosition.	After looking at the code inside this function we were sure the function updateCaretStatus that used getCaretPosition was used to populate the data in the statusBar.
11	We made caretPosition and bufferLength in updateCaretStatus function as constant and re-run the code.	Once we saw the constant values appearing in the status bar we were sure that we concept localization was correct.

Time spent (in minutes): 35

Classes and Methods inspected

1. org/gjt/sp/jedit/actions.xml
2. org/gjt/sp/jedit/textarea/JEditTextArea.java
    a. doWordCount()
    b. showWordCountDialog()
3. org/gjt/sp/jedit/textarea/TextArea.java
4. org/gjt/sp/jedit/View.java
    a. setEditPane()
    b. handleEditPaneUpdate()
    c. CaretHandler
        - caretUpdate()
    d. ScrollHandler
        - scrolledVertically()
5. org/gjt/sp/jedit/EditPane.java
    a. handleBufferUpdate()
6. org/gjt/sp/jedit/Abbrevs.java
7. org/gjt/sp/jedit/gui/StatusBar.java
    a. updateCaretStatus()

2. Impact Analysis

Step #	Description	Rationale
1	Looked into the different files and functions where the updateCaretStatus.	We listed all the functions in order to see if there is any function that affects the change we made.
2	The method to calculate the number of words was already in place.	The JEditTextArea.java had the method doWordcount to count the number of words.
3	To avoid duplication of code we decided to change the doWordcount method.	Code duplication is a bad programming practice so we chose to avoid that.
4	We analyzed the code where doWordcount and showWordCountDialog were being used.	Due to the approach for the change request, we had to analyze the impact of doWordcount and showWordCountDialog.
5	We analyzed each of the functions to understand the effect of our change.	We came to the conclusion that the change we planned would not affect any other part of the code and there was no need to change the source code any further as a result of our change

Time spent (in minutes): 25

Classes and Methods inspected

1. org/gjt/sp/jedit/EditPane.java
    a. handleBufferUpdate()
2. org/gjt/sp/jedit/View.java
    a. setEditPane()
    b. handleEditPaneUpdate()
    c. CaretHandler
        i. caretUpdate()
    d. ScrollHandler
        i. scrolledVertically()
3. org/gjt/sp/jedit/textarea/JEditTextArea.java
    a. doWordCount()
    b. showWordCountDialog()

3. Actualization

Step #	Description	Rationale
1	The initial thought was to create a new method that would compute the number of words.	This approach results in code duplication as the same method to count words is already present. We thought of reusing the same method.
2	In order to use the doWordCount method the changes are as follows: - • Convert the method to public from protected • Add a return type and return the data that we wanted • The doWordCount method is also used by the showWordCountDialog function small changes need to be made there as well.	In this way, there is no duplication of code and also maintaining the code becomes easy.
3	After our change, we ran the code and made manual few manual tests.	Made sure the changes we made did not affect other functionalities.
4	Our changed code was pushed into the respective branch.	We thought it was a good idea to push the code to a separate branch just in case.

Time spent (in minutes): 30

Classes and Methods inspected

1. org/gjt/sp/jedit/gui/StatusBar.java
    a. updateCaretStatus()
2. org/gjt/sp/jedit/textarea/JEditTextArea.java
    b. doWordCount()
    c. showWordCountDialog()

4. Validation

Step #	Description	Rationale
1	Inspection Case: After the change inspected if the changes were in place. Created a file and as we typed checked if the word count and total word count were changing. Expected Output: the word count and total word count should change as we type words.	The idea behind this is to check if the changes were in place and working as expected The test passed.
2	Test case: checking the character and line count. Inputs: opening a file with multiple lines and words. Expected output: the character counts, line, and word counts should be the same as calculated manually.	This case was to make sure that the change made did not affect character count and line count. The test passed.
3	Inspection case: on clicking the status bar where counts are displayed the go-to line dialog appears. Test functionality of dialog box. Expected output: given a line number cursor moves to that line.	This test case is more like checking if the previous functionality that was in place is still working as expected. The test case passed

Time spent (in minutes): 20

5. Summary of the change request

Phase	Time (minutes)	No. of classes inspected	No. of classes changed	No. of methods inspected	No. of methods changes
Concept location	35	7	-	5	-
Impact Analysis	25	5	-	4	-
Actualization	30	2	-	-	3
Validation	20	-	-	-	-
Total	100	14	-	9	3

6. Conclusions

The change request could have been achieved in two ways that is by adding a new method or by changing the existing method, we chose to edit the existing method as adding a new method would lead to code duplication. Due to this change in the existing we had to make sure that the existing functionality is not affected, this was done during impact analysis and testing.