Change request log

1 Team

Evolutie – Jason Stock (Developer/Document Specialist) and Kevin Bruhwiler (Developer)

2 Change Request (je1)

The left side of the status bar of jEdit reports: the line number containing the caret, the column position of the caret, the character offset of the caret from the beginning of the file, and the number of characters in the file. The request is to modify the status bar to show: the word offset of the caret from the beginning of the file and the number of words in the file. For the sake of consistency, the new caret display options should be added to the Status Bar preference dialog as check boxes.

These changes can be seen on GitHub under the **`stock-dev`** branch: https://github.com/stockeh/cs515-001-s20-evolutie-jedit

3 Concept Location

The following table describes the steps followed when performing concept location for this request. This includes various techniques, such as IDE features used, search queries, system executions, classes visited, etc.

Step #	Description	Rationale
1	Launch the current version of jEdit and inspect the Status Bar preferences to view where the options are being set. Chose to follow the logic for the caret offset.	If able to locate where the property for "showing the caret offset from start of file", then the functionality will be quite similar.
2	Used the Eclipse Search in File functionality to search for "Show caret offset from start of file" reports two matches in the workspace. One match is in the `StatusBarOptionsPane` class (default value) and the other is within the localization `jedit_*.props` (where the configurable value is set).	The location of the current status bar properties will help guide where the changes might have to take place for the word count.
3	The boolean selected value indicating the initial selection state is defined by the jEdit properties "`view.status.show-caret-offset`" This value is now used as input to search again. There are five matches with three in the `StatusBarOptionsPane` class, and two in the `StatusBar` class. We know that the `StatusBarOptionsPane` class will have to be updated to include a `JCheckBox` for the options page. We mark this class as located.	Changes for the new word offset and total number of words will need an option to be selected in the Status Bar options page. This is similar to the existing solutions and hence, why changes will have to be made to the `StatusBarOptionsPane`. We confirm this class file has to be updated.

Time spent (in minutes): 30

4 Impact Analysis

The table below describes each of the steps performed during impact analysis for this request. This includes navigating the set of classes that are related to the located class and marking it as unchanged, changed, or propagated depending on the estimated impact.

Step #	Description	Rationale
1	Following the results of searching for the property "`view.status.show-caret-offset`" from the `StatusBarOptionsPane` class in the Eclipse Search for Find in File shows the `StatusBar` class. shows where the jEdit properties for the existing caret position display are added to the buffer for the current status. We mark the `StatusBar` class as to be inspected and then to changed as a result to include the new word offset and total number of words.	The properties leverage *hidden propagation* of state between classes, and as such, require a deeper search. We confirm this class has to be changed.
2	Visited the `jedit_en.props` file within the `org.jedit.localization` package which is referenced from the `StatusBarOptionsPane` as noted in step 2 of concept location. This was found using the Eclipse IDE search functionality. This properties file was marked as to be inspected. It was clear that changes will have to occur within the properties file for the new options, and therefore we mark the properties file as located.	The new property will have to be added to control the configurable setting for jEdit. This was considered because the option is referenced from the `StatusBarOptionsPane` class. We confirm this properties file has to be changed.
3	The caret position for the what is written to the buffer in the `StatusBar` class is obtained from the a method in the `TextArea` class named `getCaretPosition()` We mark this class as to be inspected. However, after further investing it was observed that the changes to `TextArea` were not required.	It was important to check inspect if the class would have to be modified, but after further review we deemed the changes would not propagate to this class.
4	Within the `StatusBar` class there is reference to the `Buffer` of the current text area within the `updateCaretStatus()` method. The buffer was further added as to be inspected. This class looks to be beneficial for providing information of the current text but would not have to be affected by this change. Therefore, we could mark the class as inspected but not modified.	The buffer would provide useful information about the current state of the text area. However, the `Buffer` class does not have to be modified.

5	The query "wordcount" is executed in the Eclipse IDE search in file field. This returns seven matches in the `JEditTextArea` class. This class is added to the list of classes to inspect.	Prior to concluding impact analysis, an extra search is done to see if there are any references to word count methods or attributes. It is possible that the functionality could be levered if it already exists
6	There are two methods for responsible for computing a wordcount, including: `showWordCountDialog()` and the method `doWordCount()`.	It is possible that these methods could be levered for computing the word count component of this feature request.
7	Using the Eclipse IDE search functionality it is seen that the `showWordCountDialog()` method is invoked by the `actions.xml` file with `jedit_gui.props` file. Effectively, this method is invoked when the user selects the following in the GUI: Edit ⟶ Word Count… The method `doWordCount()` could not however be directly reused for this feature request. It is decided that this request will not directly relate to the functionality for the dialog and the class `JEditTextArea` is marked as visited but unchanged.	There already exists a function to compute the wordcount for the given view, but this is specific to the Edit dialog. Changes would be similar but not exact for this feature request. Therefore this class is not considered needed to change.
8	We can conclude the impact analysis.	Changes would remain local within the `StatusBar` class. Changes would only be reading information. As such, there are no propagated changes that will leak object references to other classes.

Time spent (in minutes): 90

5 Prefactoring (optional)

The table below describes each of the steps performed during perfactoring for this request. This includes any refactoring operations and objects or methods that are modified renamed or moved. It is possible that the scope of the change is small enough that no prefactoring is needed.

Step #	Description	Rationale
1		

Time spent (in minutes): 0

6 Actualization

The table below describes each of the steps performed during actualization for this request. Steps conducted during this process describe the changes to code, e.g., classes or methods added removed or modified.

Step #	Description	Rationale
1	Two new member variables were added to the `StatusBarOptionPane` class. Namely, `showWordOfset` and `showWordCount`.	These variables will be `JCheckBox`'s used to select and deselect the options within the Status Bar preference dialog of jEdit.
2	The two member variables are initialized in the `_init()` method of `StatusBarOptionPane`. The option `options.status.word.offset` is added to the `JCheckBox` for the default name of the checkbox description. Furthermore, the selected option indicating if the value is selected or not is saved in the `view.status.show-word-offset`. Similarly, these properties are added for the word count. These member variables are then added to the options panel.	Similar to the first step, these variables will be `JCheckBox`'s used to select and deselect the options within the Status Bar preference dialog of jEdit.
3	The `_save()` method is modified similar to the `_init()` method by adding the updates to the `showWordOfset` and `showWordCount` depending on the current state of button.	The properties for the word count and word offset are saved to be later read during initialization. This allows the application to save state over multiple sessions.
4	A new title property was added to the `jEdit_en.props` properties file for each of the options. At this point there is an option to select and deselect the option to show the word count and offset, but the actions are not yet implemented.	Since new properties were added as a result of including a new option within the Status Bar dialog a new title is needed.
5	The `updateCaretStatus()` method within the `StatusBar` class has been updated with the new local variables and the call to calculate the current word offset and count from the caret location. The options are only shown if the `view.status.*` properties specified in `StatusBarOptionPane` are set.	The status bar is updated when the caret is moved or new words are added to the page. Therefore, the word offset and total count must be updated.

Time spent (in minutes): 60

7 Postfactoring (optional)

The table below describes each of the steps performed during postfactoring for this request. This includes any refactoring operations and objects or methods that are modified renamed or moved after actualization. It is possible that the scope of the change is small enough that no postfactoring is needed.

Step #	Description	Rationale
1		

Time spent (in minutes): 0

8 Validation

The table below describes each of the steps performed during validation for this request. This includes steps after actualization such as testing or code inspections.

Step #	Description	Rationale
1	Manual Functional Test 1. Build and run jEdit from scratch with `$ ant clean build run` 2. Navigate to the Options pane (⌘F12) then to the Status Bar configurations. Ensure that both (a) show word offset from start of file, and (b) show number of words in file are selected (✓) 3. Within the Text Area type out any sentence with a known word count. 4. Ensure that the word counts are correct. 5. Ensure that the word location is updated as text is added to the Text Area (i.e., this number should match the word count when typing at the very end of a file). 6. Delete the last word from the Text Area and verify that the counts have been reduced by one. It is expected that the word count and the word offset update dynamically as text is added to the Text Area. This functionality is similar to the caret location. **NOTE:** a *word* is classified as a whitepace (and EOF/BOF) separated entity.	Pass. The word count and word offset need to functionally work as expected in the feature request. As a user types to the Text Area the values shall be shown. This test is good to verify that the options to hide and show the options that were implemented work as expected.
2	Manual Functional Test 1. Build and run jEdit from scratch with `$ ant clean build run` 2. Navigate to the Options pane (⌘F12) then to the Status Bar configurations. Ensure	Pass. The word offset updates as a result of the adding and removing words from the Text Area. Changes in the caret position yield updated word offsets. This test is good to verify that the word offset

	that ONLY (a) show word offset from start of file is selected (✓) 3. Within the Text Area type out any sentence with a known word count. 4. Ensure that only the word offset is displayed in the status bar with the correct value. 5. Move the caret using the arrow keys and the mouse. Verify that the word offset is updated with change in caret position. It is expected that only the word offset is displayed in the Status Bar with the correct value. NOTE: the offset of a word is only considered after the first character of that word (e.g., 100 words may exist, but if the caret is at position 0 then word offset is 0 until the position moves into the first character of that word).	functionality that was implemented works as expected.
3	Manual Functional Test 1. Build and run jEdit from scratch with `$ ant clean build run` 2. Navigate to the Options pane (⌘F12) then to the Status Bar configurations. Ensure that ONLY (b) show number of words in file are selected (✓) 3. Within the Text Area type out any sentence with a known word count. 4. Ensure that only the word count is displayed in the status bar with the correct value when words are both added and removed. It is expected that only the word count is displayed in the Status Bar with the correct value. Furthermore, this value does not change with modifications of the caret position.	Pass. The word count remains consistent with what is written in the Text Area. This value updates similar to the total number of characters in the Text Area. This test is good to verify that the word count functionality that was implemented works as expected.
4	Manual Functional Test 1. Build and run jEdit from scratch with `$ ant clean build run` 2. Navigate to the Options pane (⌘F12) then to the Status Bar configurations. Ensure that both (a) show word offset from start of file, and (b) show number of words in file are selected (✓) 3. Open a file that is more significant in size (e.g., 10KB)	Pass. This test focuses on using a larger file and validating the correctness of the new feature when moving within the Text Area.

	4. Record the word count of this file by using the command line and running the following: `$ cat file.txt | wc -w` 5. Within jEdit compare the word count of the file with that seen by running the bash **wc** command. Verify the counts match. 6. Click around the screen to move the caret position. Verify that the word offset moves as expected. It is expected that the word count is as expected and that the word offset updates with the change in caret position.	
5	Commit changes and perform code review in GitHub	Inspect changes from collaborative partner to ensure logical correction and understand

Time spent (in minutes): 60

9 Timing

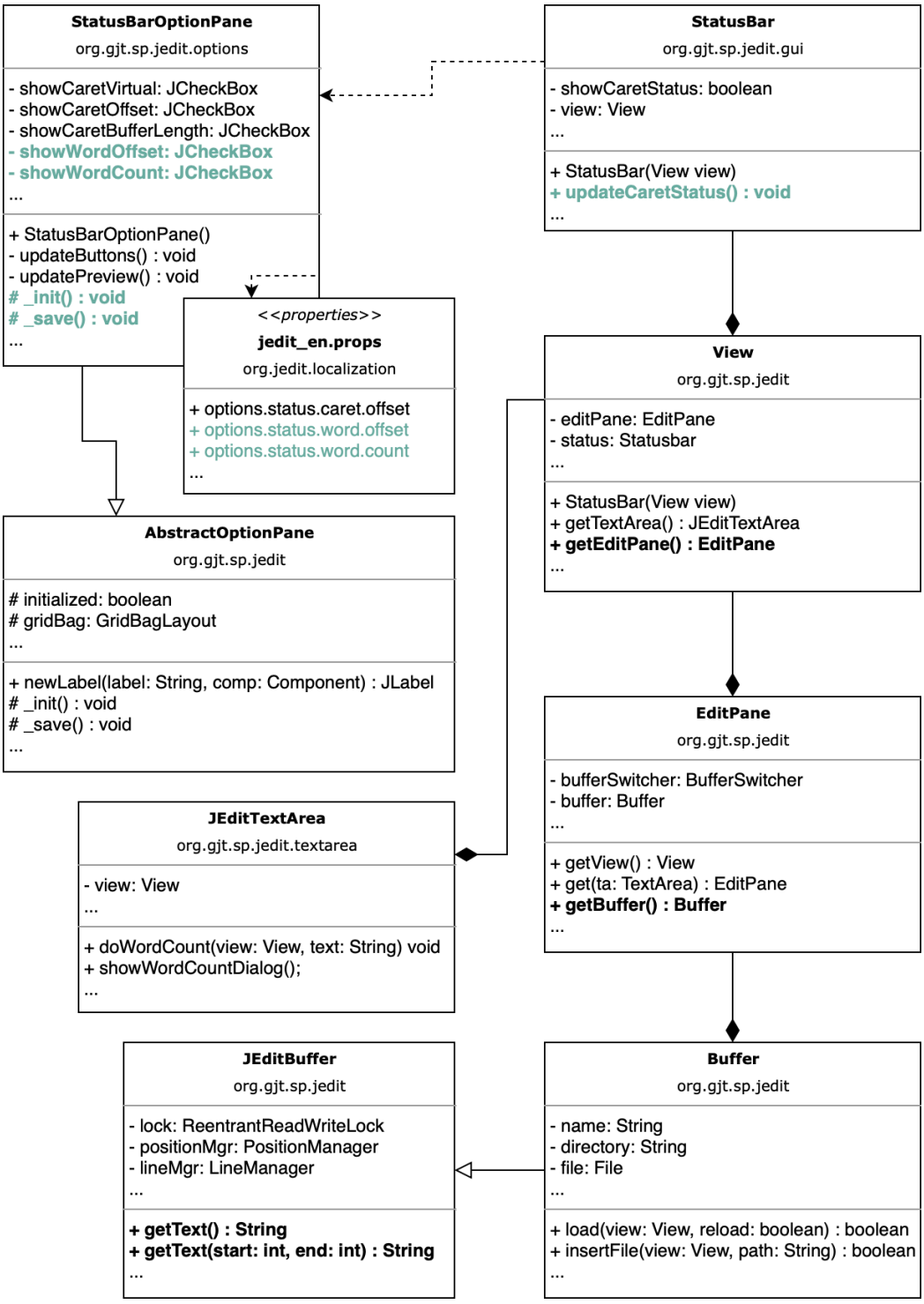
Summarize of the time spent in each step.

Phase Name	Time (in minutes)
Concept location	30
Impact Analysis	90
Prefactoring	-
Actualization	60
Postfactoring	-
Verification	60
Total	240

10 Reverse engineering

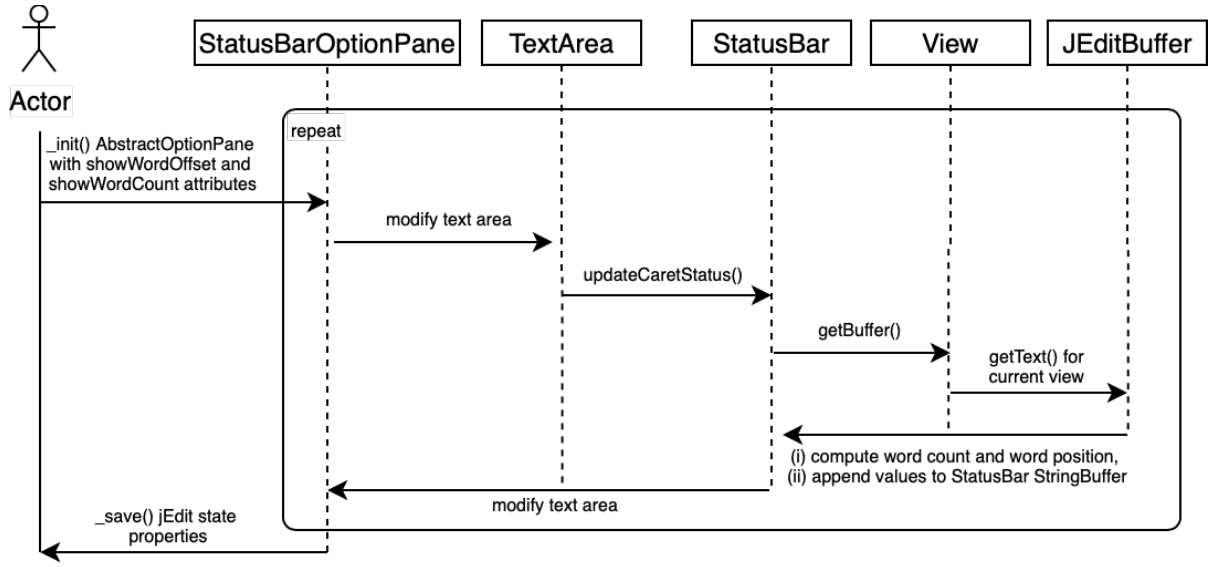
10.1 UML Class Diagram

A UML class diagram of the visited classes while navigating the code in concept location. This includes the associations between classes (e.g., inheritance, aggregations, compositions, etc.).



10.2 UML Sequence Diagram

A UML sequence diagram corresponding to the main object interactions affected by the changes in this feature.



11 Conclusions

Concept location for this feature request is seen having the least amount of time spent. This is largely influenced to completing the #je-2 feature request prior to starting this feature. As such, a mental image of the application was made and the functional components related to view options property files have been understood. Once the initial class was found the steps conducted for impact analysis consumed the majority of the time. Reason for this was due to complications around isolating which files the actual changes should be implemented in. Various considerations were made for how the word count and offset should be calculated which would influence the impact set. By understanding how the text buffers are designed the functional calculated changes related were seen to remain local within the `StatusBar` class.

The following classes and methods were updated:

- `org/gjt/sp/jedit/gui/StatusBar.java`
  - `updateCaretStatus()`
- `org/gjt/sp/jedit/options/StatusBarOptionPane.java`
  - two new instance variables added; namely, `showWordOffset` and `showWordCount`
  - `_init()`
  - `_save()`
- `org/jedit/localization/jedit_en.props`
  - `options.status.word.offset`
  - `options.status.word.count`