Change request log

1 Team

Jared & Yashwanth

Jared Crouse: Code navigation and changes

Yashwanth Virupaksha: Change request log tracker

2 Change Request

#ps2

The Merge module throws an exception upon attempting to merge page ranges that intersect (see Figure 7 and Figure 8). You are to fix this issue by allowing intersection of ranges during the merging operation.

3 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 PDFsam so we could look at the ui to help determine what may need to change*	*We figured it would help to look at the UI to help investigate what will need to change in the code.*
2	*We found the merge package in the pdfsam code base and identified two files that may need to change including MergeParametersBuilder.java and MergeSelectionPane.java.*	*We started looking in the merge module to begin identifying code within files that may need to change.*
3	*We began looking through the pdfsam-fx package for files that may be related to parsing the page ranges entered in the merge module and identified PageRangesColum.java, SelectionTableColumn.java and SelectionTableRow.java.*	*It was our thought that looking through these files would lead us to where we would actually need to make changes to the code.*

4	*We realized that some of the previously stated files were using utilities in the pdfsam-core package, we found ConversionUtils.java in the params folder within this package.*	*We are just looking at related files and dependencies to see what may need to change.*
5	*We marked the class ConversionUtils.java as "located"*	*We confirmed this class had to be modified.*

Time spent (in minutes): 126

4 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.

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

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 detail 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 looked at the ConversionUtils.java file and knew we needed to make changes to the toPageRangeSet() method.*	*After inspecting all the code we listed in the concept location section we think this is the only place that we will need to make changes to achieve the change request.*

Time spent (in minutes): 8

5 Prefactoring (optional) SKIPPED

Using the table below, describe each step you follow to prefactor the code. Include as many details as possible, including the refactoring operations used (e.g., move method, extract class, etc.) and classes/methods/fields that were modified, added, removed, renamed, etc.

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

Step #	Description	Rationale
1	*We converted the variable color into a field in class Schedule (method changeColor). We used the refactoring "extract field" from the IDE.*	*As many methods will access the color value, it is a good idea to have a field. This would reduce the number of arguments and parameters of the methods*
2	*After the previous change, we ran the unit tests corresponding to the class Schedule and also we ran the system. We went to the schedule screen.*	*We tested everything was working as before, after the refactoring.*

3	*We committed our changes with git.*	*Just in case we need to revert our changes.*
4	...	

Time spent (in minutes): -

6 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.

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

Step #	Description	Rationale
1	*We added an ArrayList to the toPageRangeSet() method in the ConversionUtils.java file in the pdfsam-core package.*	*We did this so we could make a list of each individual page that will be in the final merged document.*
2	*We also declared a set to create the page ranges to be added to the ArrayList in the toPageRangeSet() method.*	*The set is used to help determine what pages will be in the final merged document.*

Time spent (in minutes): 23

7 Postfactoring (optional) SKIPPED

Use the table below to describe each step you followed to postfactor the code. Include as many details as possible, including the refactoring operations used (e.g., move method, extract class, etc.) and classes/methods/fields that were modified, added, removed, renamed, etc.

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

Step #	Description	Rationale
1	*We converted the variable color into a field in class Schedule (method changeColor). We used the refactoring "extract field" from the IDE.*	*As many methods will access the color value, it is a good idea to have a field. This would reduce the number of arguments and parameters of the methods*
2	*After the previous change, we ran the unit tests corresponding to the class Schedule and also, we ran the system. We went to the schedule screen.*	*We tested everything was working as before, after the refactoring.*
3	*We committed and pushed our changes with git.*	*Just in case we need to revert our changes.*
4	...	

Time spent (in minutes): -

8 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.

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

Step #	Description	Rationale
1	*Test case defined: We add a document with 7 pages and specify intersecting ranges.* *Inputs: We specify the range as 1-5, 3* *Expected output: The final document should have pages 1, 2, 3, 4, 5, 3*	*We want to make sure that the merge functionality works as expected.* *The test passed.*
2	*Test case defined: We add two seperate documents, the first with 3 pages and the second with 5 pages, we specify intersecting ranges in both documents to test functionality.* *Inputs: The range for the first (3 pages) document as 1-3, 2. The second document range is specified as 2-4, 3* *Expected output: We expect the final document to have pages 1, 2, 3, 2 from the first original document followed by pages 2, 3, 4, 3 from the second original document.*	*We want to make sure that the merge functionality works as expected with multiple documents and intersecting ranges.* *The test passed.*
3	*Test case defined: We add one document with 7 pages and specify an incomplete intersection of pages.* *Inputs: We specify the intersection to be 2-4, 3-.* *Expected output: We expect that the final document should have pages 2, 3, 4, 3, 4, 5, 6, 7.*	*We want to make sure that the incomplete range specified as 3- includes pages 3 through the end of the document.* *The test passed.*
4	*Test case defined: We add a single document with 7 pages and specify an incomplete range and individual pages within that range.* *Inputs: We specify the range for the merged document to be 4-, 5, 7.* *Expected output: We expect the final document to have pages 4, 5, 6, 7, 5, 7.*	*We wanted to make sure that individual pages in an intersecting range would be added to the final document as expected.* *The test passed.*

Time spent (in minutes): 31

9 Timing

Summarize the time spent on each phase.

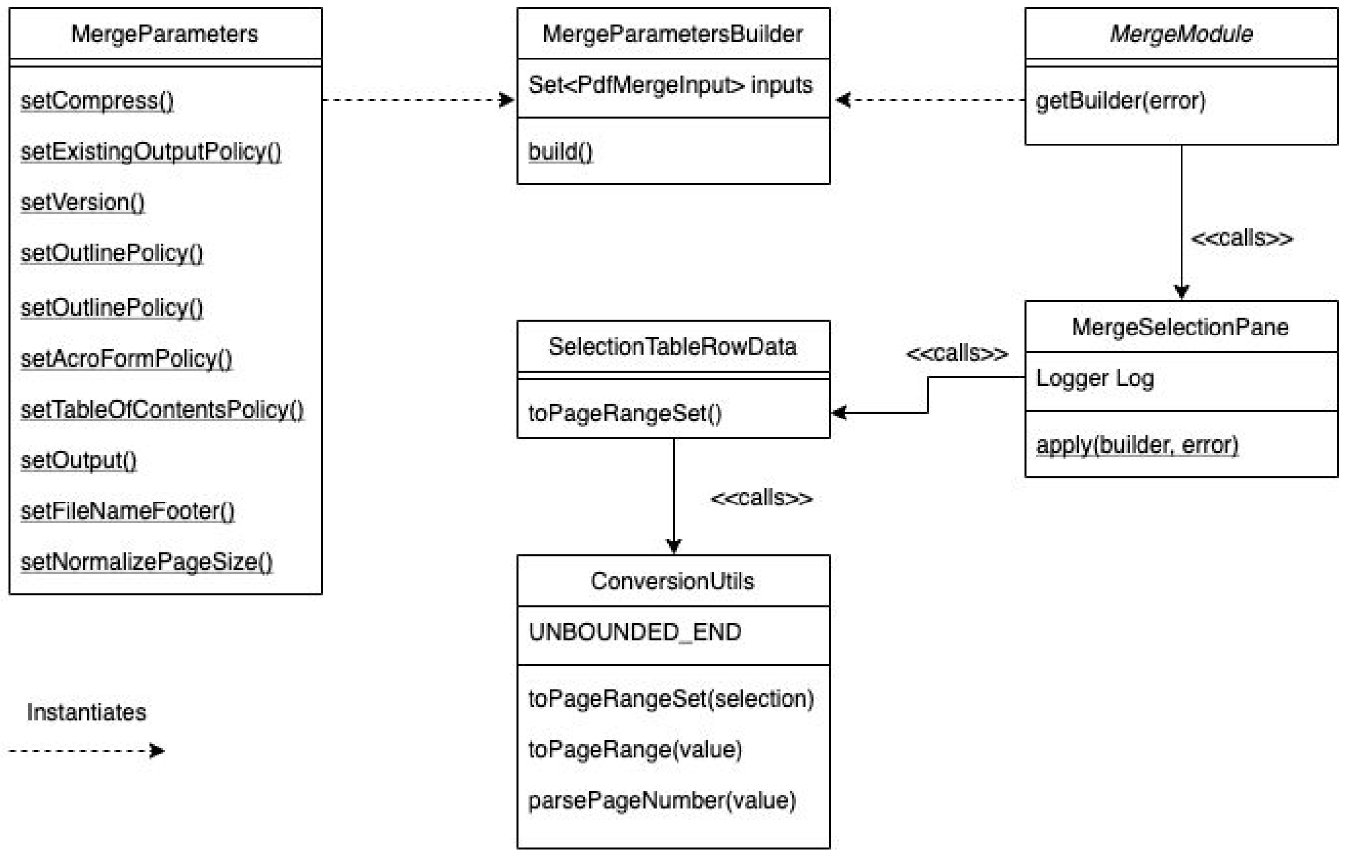
Phase Name	Time (in minutes)
Concept location	126
Impact Analysis	8
Prefactoring	-

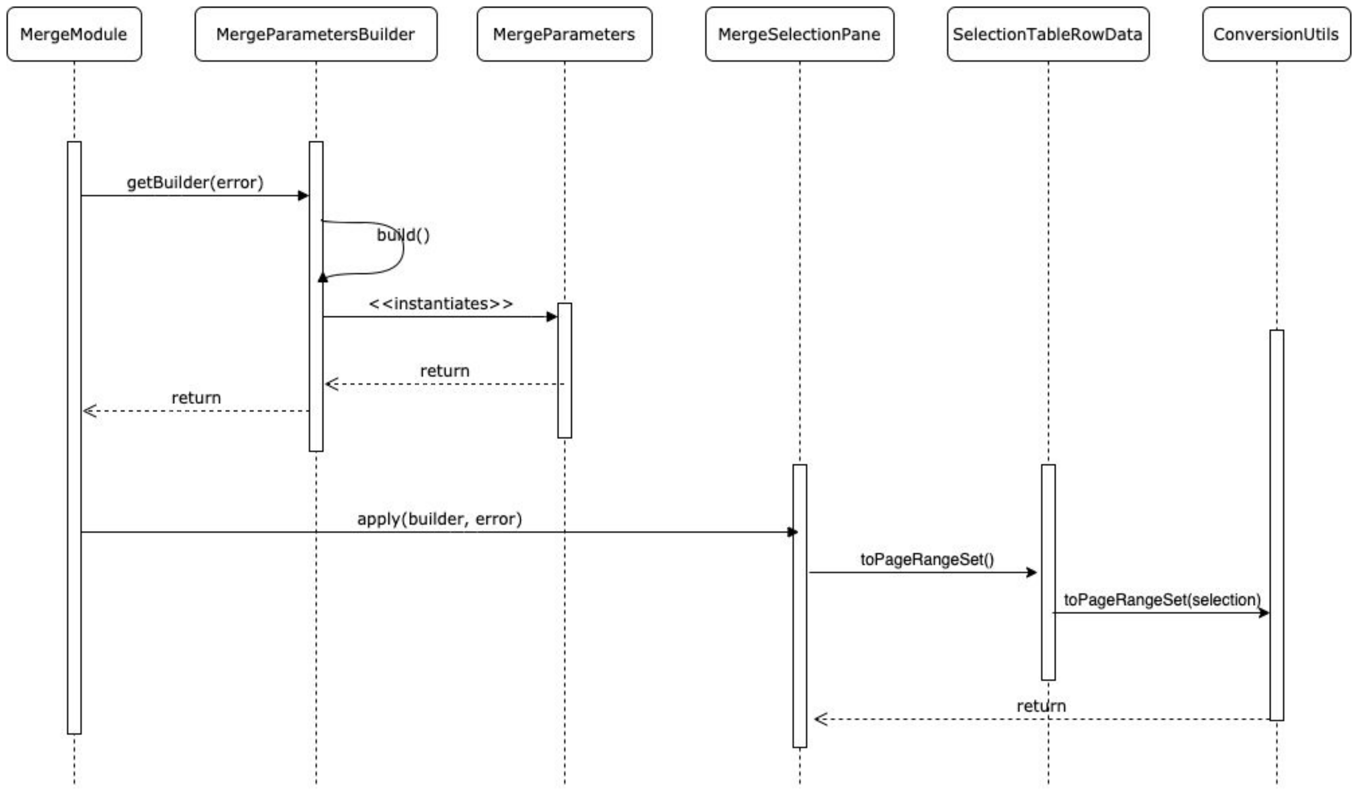
Actualization	23
Postfactoring	-
Verification	31
Total	188

10 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.**





11 Conclusions

This change request was difficult at first because PDFsam is structured very differently than jEdit which we worked on first. Concept location was once again the thing that we spent the most time on. At first we thought we would need to make changes to the sejda imports but realized that the changes would need to actually take place in the pdfsam code. Once we were able to locate the concept the impact analysis was actually fairly simple because we only needed to make changes in an existing method in the CoversionUtils.java class. Because of this, the actualization step was also not too difficult, most of the time we spent on this step was on considering how to make the change rather than making changes to the code itself. A visualization would have been helpful to do concept location but the program we used for this purpose (SourceTrail) did not support programs that need java 11 so we were not able to utilize that tool. Nevertheless we were able to accomplish the change request.

Class methods changed:

- *pdfsam-core/src/main/java/org/pdfsam/support/params/CoversionUtils.java*
    - *public static Set<PageRange> toPageRangeSet(String selection) - Changed*