Change request log

[redacted]Team

gitrams

[redacted]Change Request

Change Request ID: #3 (PDFsam)

Description of the change request: The Merge module throws an exception upon attempting to merge page ranges that intersect. You are requested to fix this issue by allowing intersection of ranges during the merging operation.

[redacted]Concept Location

- IDE Features used: **InstaSearch** for searching the relevant keywords such as 'Merge' that would lead us to functionality where merging was implemented.
- Queries used when searching: 'Merge'
- Interactions with the system: Merge page of PDFsam
- Classes visited: `MergeModule`, `MergeOptionsPane`, `MergeParametersBuilder`, `MergeSelectionPane`, `ConversionUtils`
- Approaches: We tried to implement this change request using two approaches. Finally, only one of them was found to work.
- The first class found to be changed (this is when concept location ends): `ConversionUtils`

Step #	Description	Rationale
1	*We compiled and ran the version of PDFsam from change request 1.*	*To work on latest version of PDFsam.*
2	*We interacted with merge function of PDFsam to understand the basic features that did the merging given a set of page ranges.*	*To get familiar with the existing features of merge module, that helped us understand how merging is implemented. We identified the elements we had to change and add to implement the change request.*
3	*We identified two approaches that could solve the error that was caused when we called merge module with overlapping page ranges.*	*We wanted to explore all possible ways to implement this change request.*
4	*We searched for "merge" using the InstaSearch plugin installed in Eclipse Oxygen2.*	*To find all modules that possibly implemented the merge functionality*
5	*From over 5 results outputted, we explored following classes `MergeModule`, `MergeOptionsPane`, `MergeParametersBuilder`, `MergeSelectionPane`, `ConversionUtils`*	
6	*For approach 1: We found that the error was thrown in the imported module `'org.sejda.model.input.PdfMergeInput'`*	*We wanted to see if following this approach will make it easier to implement the change request.*
7	*For approach 2: We found that the ranges we passed to the PdfMergeInput module after performing preprocessing under `ConversionUtils module.`*	*We wanted to see if following this approach will make it easier to implement the change request.*
8	*For approach 1: We decided to extend the `org.sejda.model.input.PdfMergeInput`*	*Approach 1 had fewer changes as compared to approach 2.*

	`class` in the merge package as `PdfMergeInputMod This seemed easier because we just had to remove the` @NoIntersections check from this module.	
9	*For approach 2: We had to add a method in `ConversionUtils` that will remove the overlapping pages before passing the page ranges to `PdfMergeInput module`*	*Approach 2 had more changes as compared to approach 1*

Time spent (in minutes): 250

[redacted]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.

Following impact analysis is for approach 2 that worked and solved the error because of overlapping page range sets.

Step #	**Description**	**Rationale**
1	*The class `ConversionUtils from org.pdfsam.support.params module` was a standalone class in the context of merge module.*	*We verified this using the UML diagram for the classes ConversionUtils* `and the modules org.pdfsam.support.params and org.pdfsam.merge.`
2	*Adding a method `public static Set<PageRange> toNewPageRangeSet(Set<PageRange> pageRangeSet)` did not impact any other methods in the same class as well as any other classes from the merge modules that basically implemented the merging part where error occurred.* *Except the newly added method lead to change one line in the original method `public static Set<PageRange> toPageRangeSet(String selection) throws ConversionException from the` class `ConversionUtils.`*	*This newly implemented method just changed the returned page range set form the original method mentioned on the left.*

Time spent (in minutes): 100

[redacted]Prefactoring (optional)

For this change request we did not require to prefactor any part of the code base because the merge functionality was already implemented in modular way.

Time spent (in minutes): 0

Actualization

For actualization we tried two approaches, out of which only approach 2 mentioned above worked. So, the following table describes actualization phase for approach 2.

Step #	Description	Rationale
1	*For approach 2, we added a method in* `ConversionUtils` *that will remove the overlapping pages rage sets before passing the page ranges to* `PdfMergeInput module`	*We were sure that approach 2 was definitely going to work, so we tried approach 1 first but that ended up failing.*
2	*The newly added method is:* `public static Set<PageRange> toNewPageRangeSet(Set<PageRange> pageRangeSet)` *This method, takes the overlapping page rages and removes the overlap. It returns a non-overlapping page rage set that will contain all the pages that the user has inputted in the merge module of the GUI.*	*This method will remove the overlaps that caused the error in* `org.sejda.model.input.PdfMergeInput module`
3	*We tested these changes after building and running the PDFsam software again.*	*The change request was successfully implemented.*

Time spent (in minutes): 150

Postfactoring (optional)

For this change request we did not require to postfactor any part of the code base except for adding a few comments.

Step #	Description	Rationale
1	*Added comments regarding the modified merge functionality that was implemented using approach 2 mentioned above.*	*We wanted to keep a record/documentation in the form of comments in the source code.*

Time spent (in minutes): 10

Validation

We performed manual testing of the changed functionality of merge button. We tested with several inputs. Following are the test results on some of the test cases.

Step #	Description	Rationale
1	*We performed vigorous manual testing for this change request.*	*To test the behavior of the changed system.*
1	*Test case defined: merge on [Parnas'94].pdf* *Inputs: 1-4,2-3,6-9,7-8* *Expected output: pages 1, 2, 3, 4, 6, 7, 8, 9 in the output pdf*	*We saw this regular behavior in the generated pdf file.* *The test passed.*
2	*Test case defined: merge on [Parnas'94].pdf* *Inputs: 2-4,5-6,7-9,6-9* *Expected output: pages 2, 3, 4, 5, 6, 7, 8, 9 in the output pdf*	*We observed that the output pdf has required expected pages in it. The test passed.*

Time spent (in minutes): 30

Timing

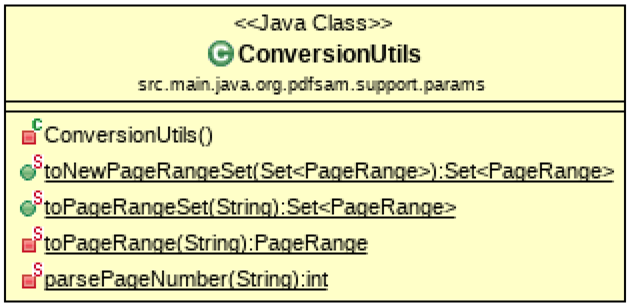
Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	250
Impact Analysis	100
Prefactoring	0
Actualization	150
Postfactoring	10
Verification	30
Total	540

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.**


<<Java Class>>
ConversionUtils
src.main.java.org.pdfsam.support.params
ConversionUtils()
toNewPageRangeSet(Set<PageRange>):Set<PageRange>
toPageRangeSet(String):Set<PageRange>
toPageRange(String):PageRange
parsePageNumber(String):int

Conclusions

For this change request, the concept location was relatively harder because the error was caused by a module that was not from the code base itself and it was imported. We identified two possible approaches to solve this error. We spent over three days (4-5 hours a day) to implement the change request using approach 1 that seemed easier and more logical. However, we could not solve the issue with approach 1 because this approach introduced some bugs that could not be solved within the deadline.

Therefore, we explored approach 2 in which the changes were only in one module `ConversionUtils` and the changes were more in terms of LoCs as compared to approach 1. This approach took over 3 to 4 hours to solve the issue in merge module regarding overlapping page ranges.

Concept location, impact analysis, actualization was done using InstaSearch plugin in Eclipse IDE.

Classes and methods changed for approach 1 that failed:

- `PdfMergeInputMod` class added in the merge module `(org.pdfsam.merge)`
- `Org.pdfsam.merge.MergeParametersBuilder`
- `Org.pdfsam.merge.MergeSelectionPane`

Classes and methods changed for approach 2 that succeeded:

- `Changed ConversionUtils class` in `org.pdfsam.support.params module`
- Added a method `public static Set<PageRange> toNewPageRangeSet(Set<PageRange> pageRangeSet)` that removes the overlapping pages ranges before calling the external module `org.sejda.model.input.PdfMergeInput.`