Change request log

1 Team

Evolutionary Rams:

1. Sanket Mehrotra
2. Nada Alalyani

2 Change Request

ID	Description
#ps-3: odd and even page ranges in the rotate module;	Apply the odd/even page filter on requested page rotations across given ranges;

3 Concept Location

In the table below we describe each step we followed when performing concept location for this change request. In our description, we include the following information when appropriate and indicate is as follows in the below table:

project

3.1 ps#3: Rotation module change request

Step #	Description	Rationale/Discovery
1	Used eclipse file search: keyword "Range". File name patterns: "Rotate*". Target: Enclosing project: pdfsam-rotate	To directly find all mentions of page ranges in the rotate module files.
2	Found 2 matching classes: RotateParametersBuilder.java (15 matches) RotateSelectionPane.java (3 matches)	
3	Checked RotateSelectionPane.java Came across PageRangeColumn() in constructor	Some mentions relate to tool-tip initialization and are present in the constructor. Notice method "builder.addInput()"
4	Open declaration of method toPageRangeSet() called in builder.addInput()	Discover class SelectionTableRowData
5	Checked RotateParametersBuilder.java	Find method build ().
6	Follow build() method to see where inputs are being handled	Discover addInput() in class RotateParametersBuilder.java
7	Find that arguments are being cast to type PdfRotationInput.	Found enum predefinedRotationType with values:

	Trace the arguments to new PdfRotationInput	ALL_PAGES EVEN_PAGES ODD_PAGES
8	In the else block of the addInput() Saw different arguments to PdfRotationInput: pageSelection.stream().toArray(PageRange[]::new)	Set breakpoint at addInput() and trace values of input page ranges in this variable.
9	Concept Location complete: First class to modify: RotateParametersBuilder present in the file RotateParametersBuilder.java	

Time spent(in minutes): 30 mins

4 Impact Analysis

Step #	Description	Rationale/Discovery
1	Any change to our finalized class and its method • Class: RotateParametersBuilder ○ public void addInput(PdfSource<?> source, Set<PageRange> pageSelection) may affect the following methods:	Checked *call hierarchy* of the addInput() function: • Class: RotateParametersBuilder ○ void apply(RotateParametersBuilder builder, Consumer<String> onError) ▪ .forEach(i -> builder.addInput(i.descriptor().toPdfFileSource(), i.toPageRangeSet())); • Class: RotateParametersBuilderTest ○ void buildDefaultSelection() ○ void buildMultiple() ○ void buildRanges() • Class: RotateSelectionPaneTest ○ void converstionException() ○ void empty() ○ void emptyByZeroPagesSelected() ○ void emptyPageSelection() ○ void notEmptyPageSelection()
2	• Similarly checking the call hierarchy of the class RotateParametersBuilder leads to :	• Class: RotateModel ○ RotateParametersBuilder getBuilder(Consumer<String> onError) • Class: RotateParametersBuilderTest ○ void setUp()
3	After listing the above classes and methods, we see that most of the references are from test classes.	It may be safe to conclude that since most of the references are not from actual functionalities, they will not significantly affect performance.
4	After reviewing the above classes in detail, we note that they all pass varying formats of PageRange objects to our target class.	We have not changed any input handling of the PageRanges, hence there is not much scope of error.

5	Our changes do not change the output as expected by these functions. The earlier code added a PdfRotationInput to the input data member of the RotationParameterBuilder class, our change preserves that function.	Hence there is no chance of breaking existing functions which rely on the output of this function i.e. the input data member of object of class RotationParameterBuilder.

Time spent (in minutes): 60

5 Actualization

5.1 ps#3:

Step #	Description	Rationale/Discovery
1	*Inside the target class addInput() we noticed two conditions were handled*	*One condition checked if there were no page ranges: apply rotation to all the document* *The else block applied the changes to the file in case page-ranges were mentioned.*
2	*In first condition a "predefinedRotationType" enum value was sent to the rotation function.*	*This enum had the value ALL_PAGES, EVEN_PAGES and ODD_PAGES.*
3	*Wrote an additional check inside the else block. This inner-if checked whether a PageRotationType(odd,even,all) had been selected by the user on the screen and which one had been selected.*	*The check had three parts, one for each Rotate-setting(ALL_PAGES, EVEN_PAGES and ODD_PAGES.).*
4	*In each checking condition, we called the same rotate code that had been called before, but added the missing inputs which had not been passed before(the even/odd page ranges or page selection)*	*The input had to be added as an array of PageRange objects to the "input" member of the RotateParameterBuilder object*

Time spent (in minutes): 60 mins

6 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.

Page Ranges were passed to test the changed functionality as (specified in the report). The following cases were checked:

Step #	Description	Rationale
1	*Test case defined: One Fixed Page range with even page rotations (rotation fixed at rotate clockwise 90 deg)* *Inputs: a pdf with 29 pages* *Expected output: The pages in the range would have their even pages rotated 90 deg clockwise*	*For an input of [2-6] even pages i.e. 2,4,6 were rotated 90 clockwise.* *This is the regular expected behavior.* *The test passed.*
2	*Test case defined: One Fixed page range with odd page rotations (rotation fixed at rotate clockwise 90 deg)* *Inputs: a pdf with 29 pages* *Expected output: The pages in the range would have their odd pages rotated 90 deg clockwise, , but the rest of the document would be unchanged.*	*For an odd input of [19-29] odd pages i.e. 19,21,23,25,27,29 were rotated 90 clockwise* *This is the regular expected behavior.* *The test passed.*
3	*Test case defined: Multiple continuous page ranges with odd/even page rotation (rotation fixed at rotate clockwise 90 deg)* *Inputs: a pdf with 29 pages* *Expected output: The pages in the range would have their odd/even pages rotated 90 deg clockwise, , but the rest of the document would be unchanged.*	*For an input of [1-13] , [19-29], all odd pages in the respective ranges were rotated 90 deg clockwise.* *This is the regular expected behavior.* *The test passed.*
4	*Test case defined: Multiple page ranges with gaps in between the ranges and odd/even page rotation (rotation fixed at rotate clockwise 90 deg)* *Inputs: a pdf with 29 pages*	*For an input of [1-13] , [19-29] and odd pages selected, all odd pages in the respective ranges were rotated 90 deg clockwise but the rest of the document was unchanged i.e. the pages stayed in their previous rotation aspects.* *This is the regular expected behavior.*

	Expected output: The pages in the ranges would have their odd/even pages rotated 90 deg clockwise, but the rest of the document would be unchanged.	*The test passed.*
5	*Test case defined: Multiple page ranges with overlaps in between the ranges and odd/even page rotation (rotation fixed at rotate clockwise 90 deg)* *Inputs: a pdf with 29 pages* *Expected output: The pages in the ranges would have their odd/even pages rotated 90 deg clockwise, but the rest of the document would be unchanged.*	*For in an input of 1-5 , 4-8 and even pages selected, all even pages i.e. 2,4,6,8 were rotated 180 degrees.* *This is the regular expected behavior.* *The test passed.*
6	*Test case defined: Rotating a file with pages already rotated 180 degrees.* *Inputs: a pdf with 29 pages* *Expected output: The pages in the ranges would have their odd/even pages rotated, but the rest of the document would be unchanged.*	*For in an input of 1-5 , 4-8 and even pages selected, all even pages i.e. 2,4,6,8 were rotated 180 degrees.* *This is the regular expected behavior.* *The test passed.*

Time spent (in minutes): 45

7 Timing

Summarize the time spent on each phase.

Phase Name	**Time (in minutes)**
Concept location	30
Impact Analysis	60
Actualization	60
Verification	45
Total	175

8 Reverse engineering

- **Class Diagram**

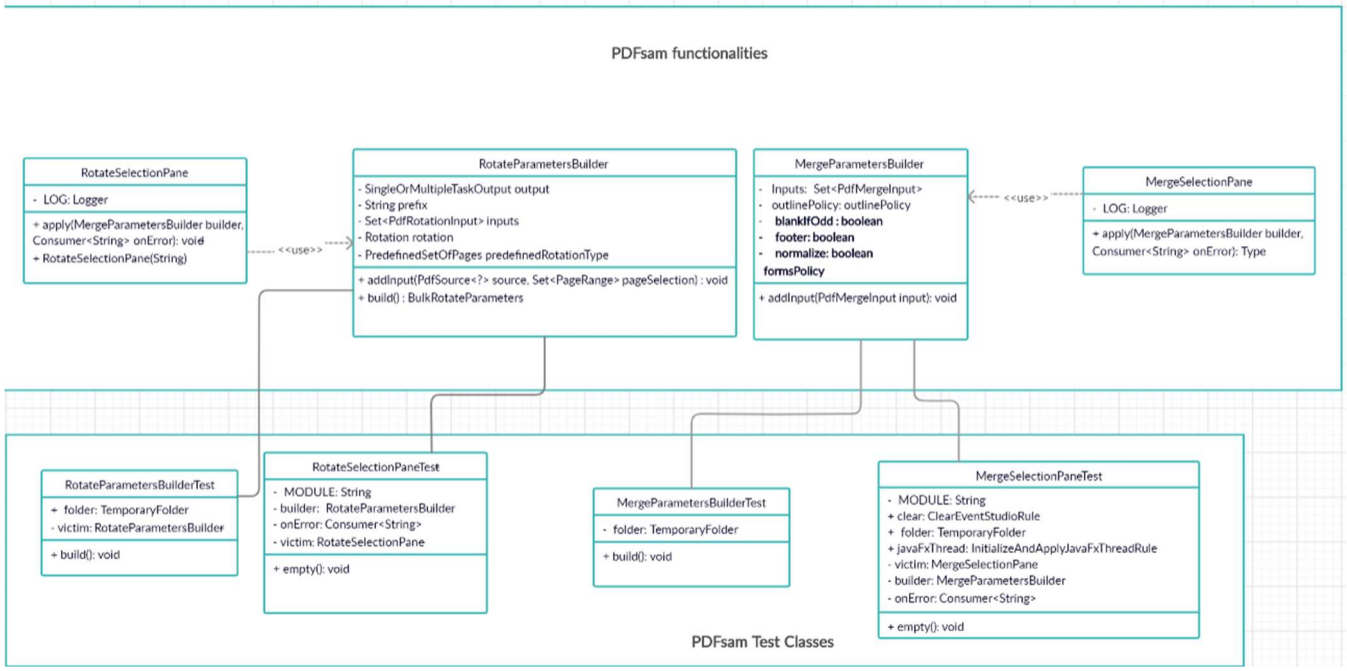

Figure 1: Class diagram for PDFsam changes ps#2 and 3

- **Sequence Diagram**

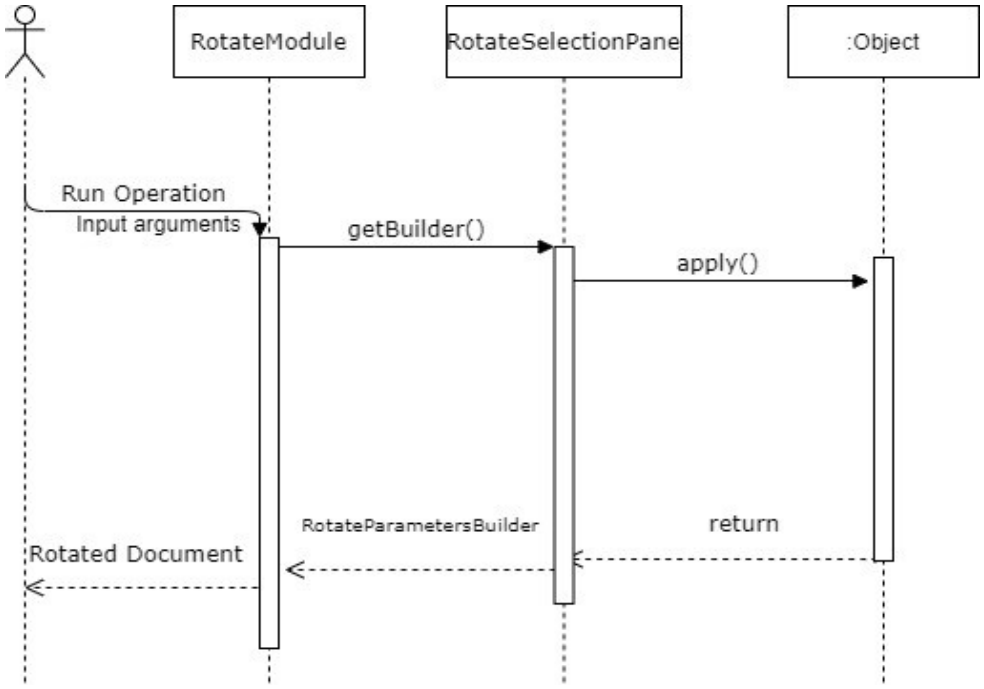

Figure 2: Sequence diagram for ps#3

9 Conclusions

This rotation change was easy to find and easier to fix. Given how much we understood of the tool in making change request #2, we didn't have to search too much before we found the concept location. Seeing how most of the code is the same for many modules in pdfsam, impact analysis also took on a similar format. Validation and actualization was done carefully. It was easier to understand than the very vague instructions in ps# 2.

Classes and methods changed:

- org/pdfsam/Rotate/RotateParametersBuilder.java
    - public void addInput(PdfSource<?> source, Set<PageRange> pageSelection)