Change request log

1 Team - bughunters

Nishant Kashiv – change request implementation (complete process)
Saurabh Deotale – documentation

Code repository - https://github.com/sd-26/cs515-001-s20-bughunters-pdfsam
(For looking up changes, go to pull requests section for the repository and search with following query: is:pr head:changeRequest#ps2)

2 Change Request

Change Request id: ps2

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

3 Concept Location

Before performing the following steps we made some changes in the build file to make sure the project builds, and the built application runs without any issues.

IDE used – IntelliJ IDEA

The following are the steps followed for concept location for change request ps2:

Step #	Description	Rationale
1	Run the installer version of the software to observe the behavior related to the change request.	To get familiar with the system and its behavior.
2	Run multiple combinations of ranges for one or more input file.	To get an idea of the cases that need to be handled.
3	● Observe the dependency graph to analyze the modules that might be involved in the complete merge workflow. ● IDE Feature used: Dependency Graph for modules. ● Possible modules: ➢ pdfsam-basic ➢ pdfsam-core ➢ pdfsam-service ➢ pdfsam-gui/pdfsam-fx ➢ pdfsam-merge (identified as the most important module based on the knowledge of software design)	Try to identify modules involved by studying the software design.
4	● Navigate and understand the modules listed above to identify possible breakpoints. ● Try to locate the module responsible to start the application. ➢ We opened the executable shell script for starting the application to find this module.	Some code navigation as preparation for debugging the application.

	➢ It's the pdfsam-basic module, which is called to start the application. ● There is a 'Run' button in the UI module 'Merge', try to locate function implementing the actions of the run button. ➢ We used the 'Find in path' functionality to search for the run button. ➢ Query was just 'runbutton', which pointed to a **Footer class in pdfsam-fx module.** ➢ Now we analyzed the usage of this class. We found that this class is used in the **'MergeModule' class** (**line 163**). ➢ Another usage is in the **'BaseTaskExecutionModule' class** (**line 63**) in the pdfsam-fx module. This class has a event registered for run button. This section was used as a breakpoint.	
5	● Debug the application to identify all the components involved in the merge module. ● The major finding was the use of Sejda SDK for pdf manipulation. ● The sejda SDK is provided with all the inputs , and that's all the code in pdfsam modules is responsible for. ● In this step we found that the merge module is responsible for collecting and formatting all the input needed for the operation, which is passed to the sejda SDK ● The workflow for merge module is as follows: ➢ We open the merge module in UI. ➢ We enter all the information needed for the operation, along with preferred settings. ➢ We click on 'Run' button. ➢ This triggers the 'MergeParametersBuilder' class's build method (line 100) through the event mapped to the run button. ➢ In this build method the input is formatted for the sejda SDK.	Try to understand the workflow implemented for the merge module.
6	● The previous step lead to one experiment with input files: ➢ When selecting files in the merge module, add the same input file twice. ➢ Now add a range for one instance of the input and another intersecting range for the second instance of same input. ➢ When we click on Run, the operation is completed and the result file has pages from both ranges, which are intersecting (this means there are duplicate pages). ➢ **This implies that if we have multiple intersecting ranges for one input, we can**	Try to find an approach to implement the change request.

	avoid the error if we can pass same input file multiple times with just one range as input for page range. So we pass an input file n times as a new input with just 1 range, where n is the number of ranges received initially. ➢ This can be done by modifying the method 'addInput' (line 53) in MergeParametersBuilder class. ➢ This method is responsible for addition of each input file in the input parameters list for sejda SDK.	
7	We identified the class MergeParametersBuilder as the root-cause.	We confirmed this class had to be modified.

Time spent (in minutes): 120

4 Impact Analysis

The impact analysis for this change request involved multiple runs of debugging the whole merge workflow. We found that once input parameters were formatted for sejda SDK they were just propagated through classes till the point they were passed to the sejda SDK task.

Step #	Description	Rationale
1	● We tried to find the usages of MergeParametersBuilder class and its build method in all modules by using the 'Find usage' functionality of the IDE. ● We found no classes having direct impact due to identified change.	Track all the classes impacted.
2	● We ran debugging operation again to follow the propagation. ● We found that it is propagated to the point where it is passed to the sejda SDK.	Track all the classes to be changed.

Time spent (in minutes): 60

5 Prefactoring (optional)

In this step we identified that the only change needed was one code snippet that can convert one input with multiple range to a list of inputs each with one range.

Step #	Description	Rationale
1	We expanded the 'addInput' method in the MergeParametersBuilder class to handle the case where one input file had multiple page ranges.	As the input and output to the method has no change, and the change introduced does not add to the atomic responsibility of the method, we can expand the method.

Time spent (in minutes): 10

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	● In the addInput' method in the MergeParametersBuilder class we added an if condition to check if there were multiple page ranges associated with each input file. ● If so, it will separately add that input file, as many times as the number of page ranges, with one page range each time.	The addInput method should handle the case with multiple page ranges (may or may not be intersecting) as previously with added condition of allowing intersecting ranges.
2	We checked if any of the unit test cases needed to be changed or if new unit test could be added.	Check if implemented unit tests need any change.

Time spent (in minutes): 20

7 Postfactoring (optional)

As the change made was contained to just one method, also the change was very short there was no need of postfactoring.

Time spent (in minutes): 0

8 Validation

We performed manual testing for this change request.

Step #	Description	Rationale
1	Test case defined: One input file with intersecting ranges as well as non-intersecting ranges Inputs: PDF file (50 pages), [5-10,11-20,15-25,35-50] Expected output:Output file with all the pages in each page-range input	The test passed.
2	Test case defined: Multiple input files with intersecting ranges as well as non-intersecting ranges Inputs: PDF file1 (50 pages), [5-10,11-20,15-25,35-50] PDF file2 (30 pages), [1-10,5-10,11-15,16-25,20-30] Expected output:Output file with all the pages in each page-range input for both the input files	The test passed.

Time spent (in minutes): 15

9 Timing

Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	120
Impact Analysis	60
Prefactoring	10
Actualization	20
Postfactoring	0
Verification	15
Total	225

10 Reverse engineering

The UML sequence diagram can be seen below. It contains some of the classes involved in the workflow.

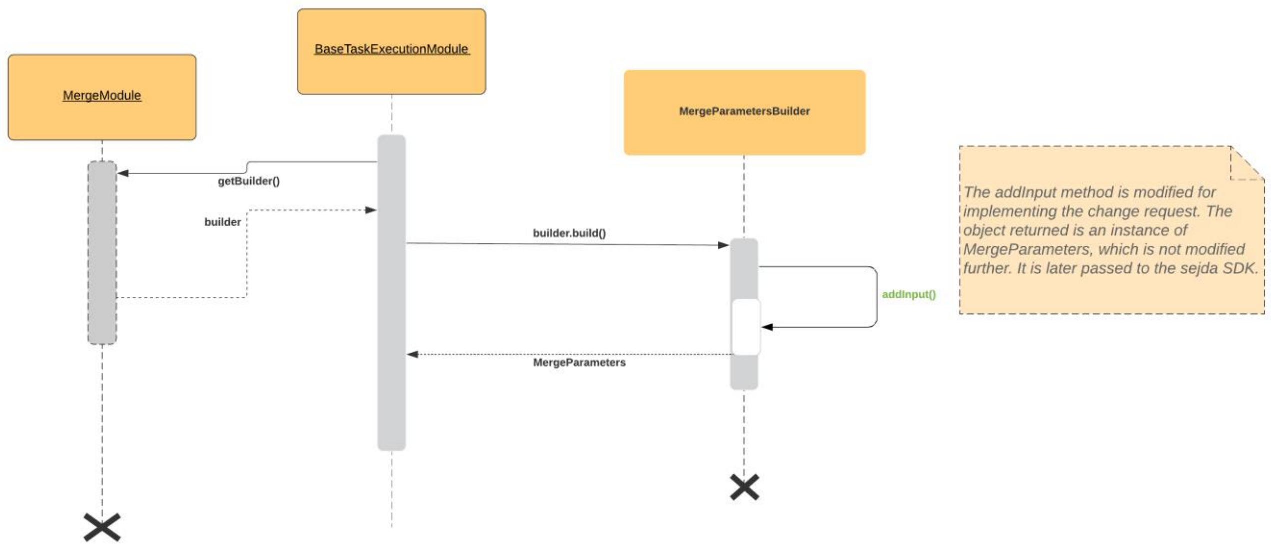
MergeModule
BaseTaskExecutionModule
MergeParametersBuilder
getBuilder()
builder
builder.build()
addInput()
MergeParameters
The addInput method is modified for implementing the change request. The object returned is an instance of MergeParameters, which is not modified further. It is later passed to the sejda SDK.

Diagram 1:UML Sequence diagram

The UML class diagram (partial) for the change can be seen below. (next page)

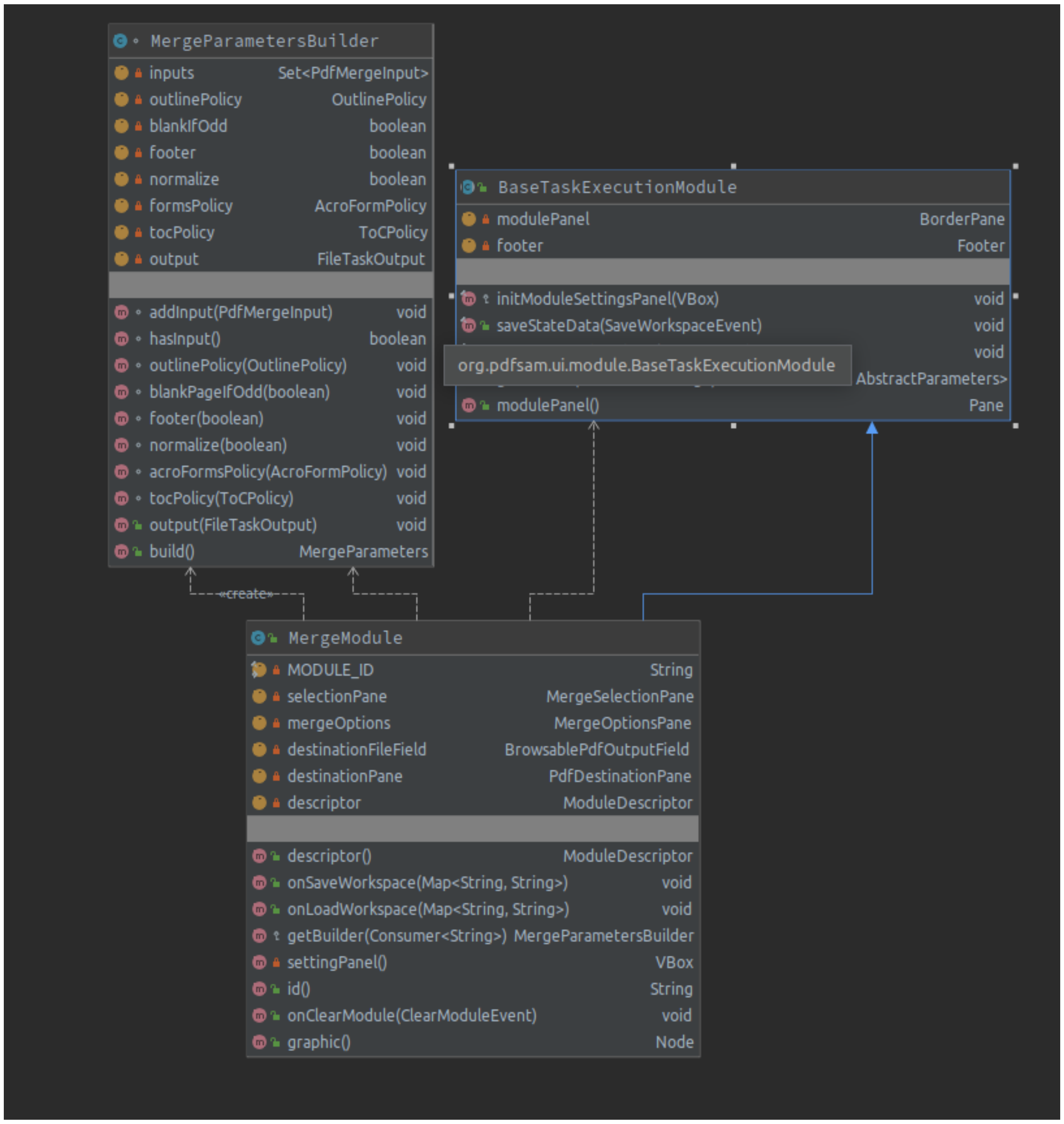
MergeParametersBuilder
inputs Set<PdfMergeInput>
outlinePolicy OutlinePolicy
blankIfOdd boolean
footer boolean
normalize boolean
formsPolicy AcroFormPolicy
tocPolicy TocPolicy
output FileTaskOutput
addInput(PdfMergeInput) void
hasInput() boolean
outlinePolicy(OutlinePolicy) void
blankPageIfOdd(boolean) void
footer(boolean) void
normalize(boolean) void
acroFormsPolicy(AcroFormPolicy) void
tocPolicy(TocPolicy) void
output(FileTaskOutput) void
build() MergeParameters
BaseTaskExecutionModule
modulePanel BorderPane
footer Footer
initModuleSettingsPanel(VBox) void
saveStateData(SaveWorkspaceEvent) void
void
org.pdfsam.ui.module.BaseTaskExecutionModule
AbstractParameters>
modulePanel() Pane
«create»
MergeModule
MODULE_ID String
selectionPane MergeSelectionPane
mergeOptions MergeOptionsPane
destinationFileField BrowsablePdfOutputField
destinationPane PdfDestinationPane
descriptor ModuleDescriptor
descriptor() ModuleDescriptor
onSaveWorkspace(Map<String, String>) void
onLoadWorkspace(Map<String, String>) void
getBuilder(Consumer<String>) MergeParametersBuilder
settingPanel() VBox
id() String
onClearModule(ClearModuleEvent) void
graphic() Node

Diagram 2: UML class diagram (partial)

The class diagram was created using IDE and the sequence diagram was created using lucidchart (online tool).

11 Conclusions

For this change concept location was a bit easier, as much of the work was already done for th previous change request. It was important to find the workflow where same file can be given as input multiple times, with each input having different range(s). And these ranges could intersect across different instances of input.

Class and method changed:

- /pdfsam-merge/src/main/java/org/pdfsam/merge/MergeParametersBuilder
    - void addInput(PdfMergeInput input)