Change request log

1 Team

Evolutie – Kevin Bruhwiler (Document Specialist, Developer) and Jason Stock (Concept Location Assistant)

2 Change Request (#ps2)

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.

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

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	Opened the project in Eclipse	
2	Went to the MergeModule class within the `java.org.pdfsam.merge` package. This class is the entry point for the merge module and responsible for the GUI.	Making modifications to PDFsam's merge functionality. Assumed MergeModule is the entry point for the package.
3	Navigated within the `java.org.pdfsam.merge` package to the MergeParametersBuilder class, which is responsible for assembling the MergeParameters.	Observed that the getBuilder(Consumer<String> onError) method applied "mergeOptions" and "selectionPane" to the MergeParametersBuilder. Both seemed likely to be involved in determining the range of pages merged.
4	Inspected the PdfMergeInput and PageRange source code from the sejda package in both Eclipse and on GitHub.	The MergeParametersBuilder class was manipulating PdfMergeInput objects when assembling the MergeParameters
5	Experimented with what caused the exception in question to be thrown. Discovered that, while ranges could not be overlapped, single pages could be repeated without issue. E.x. "1-3,2" would not work but "1,2,3,2" would.	After observing the the exception in question was being thrown by sejda, a package which we could not directly modify, we began searching for ways to circumvent the problem from within PDFsam.
6	Determined that the optimal solution would be to divide all PdfMergeInputs containing page ranges into multiple PdfMergeInputs containing only a single page within the MergeParametersBuilder class.	Sejda cannot be directly altered, so the problem must be transformed within PDFsam into something that sejda can handle.

Time spent (in minutes): 72 minutes

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	Added MergeModule, MergeOptionsPane, and MergeSelectionPane to the impact set.	Because the only method being modified is within a private method of a class that is not inherited from, impact is constrained purely to the `java.org.pdfsam.merge package.`
2	Marked MergeModule as unchanged.	MergeModule is responsible for orchestrating many operations in the the merge package, however it does not directly interact with sejda in the getBuilder(Consumer<String> onError) method.
3	Marked MergeSelectionPane as unchanged.	The MergeSelectionPane is responsible for reading the selection, filtering it, and passing it to the MergeParamtersBuilder, however we do not need to change the input that the MergeParametersBuilder receives.
4	Marked MergeOptionsPane as unchanged.	The MergeOptionsPane only interacts with the MergeParametersBuilder in the apply() method, which is stateless, has no return value, and never calls the MergeParametersBuilder's apply() method.

Time spent (in minutes): 20

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.

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

Step #	Description	Rationale
1		

Time spent (in minutes): x

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	Added functionality to the MergeParametersBuilder.addInput(PdfMergeInput input) method that splits each PdfMergeInput containing more than 1 page into multiple PdfMergeInput's containing a single page each, added all to MergeParametersBuilder.inputs	During concept location, we noticed that sejda could handle overlapping ranges when they were supplied as a list of single pages rather than an entire range.
2	Changed PdfMergeBuilder.inputs from a Set to a List. The Set could not contain duplicates and consequently ignored all but the first occurence of overlapping ranges.	When providing overlapping ranges, a user expects those ranges to occur where they were entered, not for all but the first range to be ignored.
3	Added statement to insert empty page range into PdfMergeBuilder.inputs if the provided PdfMergeInput is empty.	Several tests provide empty PdfMergeInputs to the PdfMergeBuilder and expect it PdfMergeBuilder.inputs to contain empty ranges, rather than ignoring them.

Time spent (in minutes): 85

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): x

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 the project with Maven 2. Open the Merge menu 3. Add in two PDF's, as shown in the feature request 4. Input the page ranges provided in the feature request 5. Merge the documents, do not overwrite existing files	Overlapping ranges should not throw an exception and should generate a document with those ranges.

	6. Verify that the merged document contains all those page ranges. It is expected that the generated PDF contains all the pages in the all the ranges provided from both documents	
2	Manual functional test 1. Build and run the project with Maven 2. Open the Merge menu 3. Add a single PDF document 4. Input 3 page ranges, each capturing every page of the document 5. Merge the documents, overwriting any existing files 6. Verify that the merged document contains the same pages, repeated three times each in order. It is expected that the document has been replicated three times in order.	Page ranges that show up more than once should still be merged appropriately, in order.
3	Automatic Test 1. Build and run the project with Maven, enabling all unit tests 2. Verify that all tests have passed. It is expected that all unit tests written for PDFsam still pass. Any failed tests suggest this feature has broken some key part of PDFsam's functionality.	Unit tests are assumed to have been added because they test important functionality. Failed tests suggest a serious issue.

Time spent (in minutes): 43

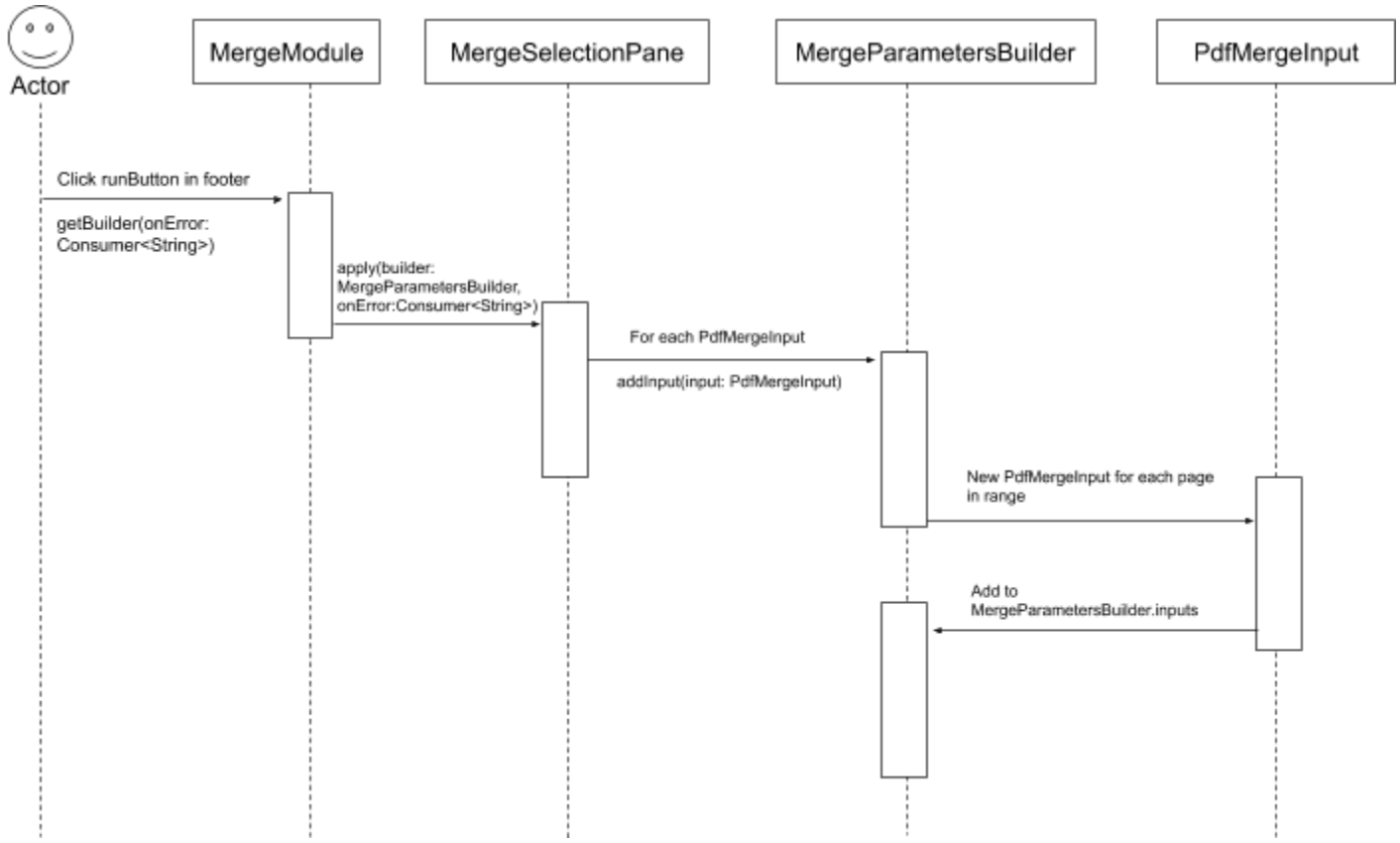
9 Timing

Summarize the time spent on each phase.

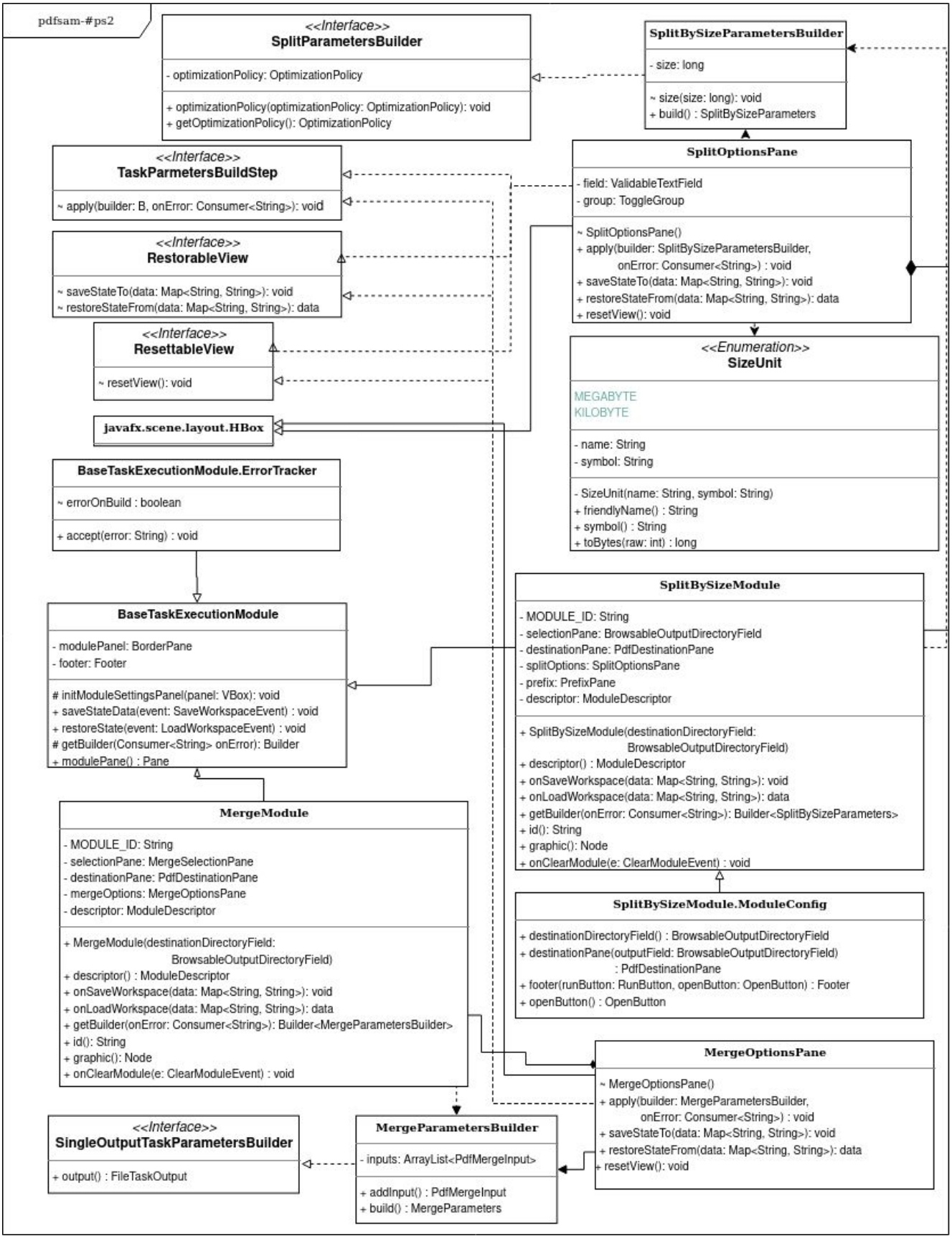
Phase Name	Time (in minutes)
Concept location	72
Impact Analysis	20
Prefactoring	-
Actualization	85
Postfactoring	-
Verification	43
Total	220 (3.667 hours)

10 Reverse engineering

Create a UML sequence diagram (or more if needed) corresponding to the main object interactions affected by your change.



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.).


pdfsam-#ps2
<<Interface>>
SplitParametersBuilder
- optimizationPolicy: OptimizationPolicy
+ optimizationPolicy(optimizationPolicy: OptimizationPolicy): void
+ getOptimizationPolicy(): OptimizationPolicy
SplitBySizeParametersBuilder
- size: long
~ size(size: long): void
+ build() : SplitBySizeParameters
<<Interface>>
TaskParmetersBuildStep
~ apply(builder: B, onError: Consumer<String>): void
SplitOptionsPane
- field: ValidableTextField
- group: ToggleGroup
~ SplitOptionsPane()
+ apply(builder: SplitBySizeParametersBuilder, onError: Consumer<String>) : void
+ saveStateTo(data: Map<String, String>): void
+ restoreStateFrom(data: Map<String, String>): data
+ resetView(): void
<<Interface>>
RestorableView
~ saveStateTo(data: Map<String, String>): void
~ restoreStateFrom(data: Map<String, String>): data
<<Interface>>
ResettableView
~ resetView(): void
<<Enumeration>>
SizeUnit
MEGABYTE
KILOBYTE
- name: String
- symbol: String
- SizeUnit(name: String, symbol: String)
+ friendlyName() : String
+ symbol() : String
+ toBytes(raw: int) : long
javafx.scene.layout.HBox
BaseTaskExecutionModule.ErrorTracker
~ errorOnBuild : boolean
+ accept(error: String) : void
BaseTaskExecutionModule
- modulePanel: BorderPane
- footer: Footer
initModuleSettingsPanel(panel: VBox): void
+ saveStateData(event: SaveWorkspaceEvent) : void
+ restoreState(event: LoadWorkspaceEvent) : void
# getBuilder(Consumer<String> onError): Builder
+ modulePane() : Pane
SplitBySizeModule
- MODULE_ID: String
- selectionPane: BrowsableOutputDirectoryField
- destinationPane: PdfDestinationPane
- splitOptions: SplitOptionsPane
- prefix: PrefixPane
- descriptor: ModuleDescriptor
+ SplitBySizeModule(destinationDirectoryField: BrowsableOutputDirectoryField)
+ descriptor() : ModuleDescriptor
+ onSaveWorkspace(data: Map<String, String>): void
+ onLoadWorkspace(data: Map<String, String>): data
+ getBuilder(onError: Consumer<String>): Builder<SplitBySizeParameters>
+ id(): String
+ graphic(): Node
+ onClearModule(e: ClearModuleEvent) : void
MergeModule
- MODULE_ID: String
- selectionPane: MergeSelectionPane
- destinationPane: PdfDestinationPane
- mergeOptions: MergeOptionsPane
- descriptor: ModuleDescriptor
+ MergeModule(destinationDirectoryField: BrowsableOutputDirectoryField)
+ descriptor() : ModuleDescriptor
+ onSaveWorkspace(data: Map<String, String>): void
+ onLoadWorkspace(data: Map<String, String>): data
+ getBuilder(onError: Consumer<String>): Builder<MergeParametersBuilder>
+ id(): String
+ graphic(): Node
+ onClearModule(e: ClearModuleEvent) : void
SplitBySizeModule.ModuleConfig
+ destinationDirectoryField() : BrowsableOutputDirectoryField
+ destinationPane(outputField: BrowsableOutputDirectoryField) : PdfDestinationPane
+ footer(runButton: RunButton, openButton: OpenButton) : Footer
+ openButton() : OpenButton
MergeOptionsPane
~ MergeOptionsPane()
+ apply(builder: MergeParametersBuilder, onError: Consumer<String>) : void
+ saveStateTo(data: Map<String, String>): void
+ restoreStateFrom(data: Map<String, String>): data
+ resetView(): void
<<Interface>>
SingleOutputTaskParametersBuilder
+ output() : FileTaskOutput
MergeParametersBuilder
- inputs: ArrayList<PdfMergeInput>
+ addInput() : PdfMergeInput
+ build() : MergeParameters

11 Conclusions

The main challenge in resolving this feature request was in determining how to deal with it without changing the behavior of the sejda package. Fortunately, we discovered that sejda could handle overlapping ranges if they were provided as individual pages rather than an entire range. There was also some difficulty in getting all the unit tests to pass, as a number of the org.pdfsam.merge tests depended on the merge parameters containing empty page ranges, a behavior that I found counter-intuitive.

However, once the issues above were resolved the feature proved relatively straightforward to add, as only a single, private method needed to be modified. The only additional hiccup was the fact that the MergeParametersBuilder used a Set to store its inputs, meaning that duplicate single-pages would be ignored because the Set could not contain duplicates. Changing the Set to an ArrayList, which does allow duplicates, resolved that issue.

Classes and methods changed:

- org.pdfsam.merge.MergeParametersBuilder
    - void addInput(PdfMergeInput input)