Change request log

1 Team

Evolutie – Jason Stock (Developer/Document Specialist) and Kevin Bruhwiler (Developer)

2 Change Request (ps1)

The Split by size module of PDFsam allows to split a PDF file in files of a given size. Ideally, the created files should be smaller than the given size, but this does not always happen. You are requested to fix this functionality, so that the created files never exceed the specified size, unless the created file contains a single page that by itself exceeds the limit size.

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	Ran the program and tried to replicate the results seen for "Split by Size". The scenario could be replicated.	Need to ensure the issue could be replicated.
2	Navigate to the `pdfsam-split-by-size` module and inspect the five classes under `org.pdfsam.splitbysize` package (this is the only package within the module).	This module is responsible for specifying the parameters, and options for the Split by Size module and should provide some information.
3	`SplitBySizeParametersBuilder` class looks to be a location for where the parameters are saved specifically for this Split by Size operation.	State needs to be maintained within each operation, and the `*ParamatersBuilder` helps to maintain this state among each module.
4	The '`size`' member variable within `SplitBySizeParametersBuilder` is set via the `size()` method. This was followed with the Call Hierarchy in Eclipse, and four results were returned. Three of which were the jUnit test classes.	The size for the split is likely set with the `size()` method.
5	Followed the result of the Call Hierarchy to the `apply()` method within the `SplitOptionsPane` class. Using the Call Hierarchy for this method shows the method being called by one non-test method named `getBuilder()` within the `SplitBySizeModule` class. This method extends the `BaseTaskExecutionModule` and is	If it could be understood how the run button executes, then it could be better understood how the size of the split is actually set. Therefore, by following the steps of how the size is set back to where the run button is executed, then we could evaluate how the size is used.

	called when the run button sees an action and is fired according to the `EventHandler` within the `initModuleSettingsPanel()` method of the `BaseTaskExecutionModule` class.	
6	Enabled `DEBUG` Logging levels with log4j for the Sejda library. Ran the application again from within Eclipse to observe the console logs. Split a document of size 2,402,750 bytes at 700 kilobytes.	Due to the fine level of logging in PDFsam and Sedja the logs should include more information for how the application actually function. This is because Sedja is the open source SDK to modify the PDF.
7	It was noticed that the actual split size used is `DEBUG 12:32:45.284 o.s.i.s.SplitBySizeTask[pool-2-thread-1] Starting split by size 716.80 KB` This is not 700 kilobytes but 700 kibibytes, i.e., 716.80 / 1.024 = 700 This results in a given split of the file on disk to have a of size 701,707 bytes (701.707 KB or 685.2607 KiB) as viewed in the file explorer.	The split size is not consistent with the values specified in the GUI.
8	In `apply()` method of the `SplitOptionsPane` the size is converted to bytes using the `toBytes()` method in the `SizeUnit` enumerator. Following the declaration in Eclipse, it is seen that the `toBytes()` method is seen falsely converting the kilobytes to kibibytes. The change required for this feature will occur within the `SizeUnit` enumerator – marking the class as located.	The sizes are falsely represented and should instead be consistent with use of kilobytes.

Time spent (in minutes): 140

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	Made a list of classes that reference the enumerated fields from within `SizeUnit`. This was done using the Eclipse IDE Call Hierarchy functionality to see what classes utilize the methods and fields of the class.	A list for classes that reference the instances of the `SizeUnit` should be considered.

	This class has no imported state variables, i.e., all members reside within the class.	
2	There are no main classes that depend on the values of the enumerator directly. Specifically, there are no conditions that state that rely on the specific values. This could be observed by tracing the usage of the methods and values returned by the Call Hierarchy.	Tracing the location of when/where/how the `SizeUnit` values and methods are used will determine which classes are impacted by the change to the enumerator values.
3	There are a number of jUnit `*Test` classes that rely on the value returned by the enumerator. These classes do not affect the functionality of the application but determine if the application can successfully be built.	Test classes under `src/test/java` do not dictate the functionality of the application but are importation to have for future maintenance and evolution of the application.
4	There are no other classes that have to be modified other than the jUnit Test classes and the enumerator.	

Time spent (in minutes): 40

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.

Step #	Description	Rationale
1		

Time spent (in minutes): 0

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	Opened the `SizeUnit` enumerator to edit and changed two lines of code to correctly identify the value of a kilobyte instead of kibibytes.	Need to have the correct conversions to kilobyte.
2	Modified the jUnit tests in the `SizeUnitTest` and `SplitOptionsPaneTest` classes.	Need to ensure the application runs as expected.
3	Commit the change to GitHub.	In case the change had to be rolled back.

Time spent (in minutes): 15

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): 0

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. Compile and build the application 2. Run the `pdfsam.sh` script within the basic target module 3. Navigate to the "Split by Size" module within the GUI and upload a large PDF file (e.g., Simba-GEAR1.pdf 2,402,750 bytes) 4. Test a range of split values in 100KB intervals between 100 – 1000 KB and ensure a previously selected size no longer violates the feature. 5. Verify the output file created adheres to the specification of the request (i.e., generated files are smaller than the given size except for single page files). It is expected that all files should be less than the requested size specifications.	Pass. The input file which previously created 8 files where two splits (split 1 and split 10) had size greater than that specified within PDFsam was used again for this test. Using this same file and a range of split sizes the results were seen with output files less than that specified unless they were single page output.
2	Manual Functional Test 1. Set the log4j Logging level to `DEBUG` within the `logback.xml` file. 2. Run the application and navigate to the "Split by Size" module. 3. Load the file that was not previously split correctly (e.g., the previously failed split had a size of 700KB. This value is reused) 4. Split the file using a size of 700KB	Pass. It could be seen that the log output from Sedja is `DEBUG 13:34:13.635 o.s.i.s.SplitBySizeTask[pool-3-thread-1] Starting split by size 700.00 KB` where the input is actually being split on 700 KB sizes. The resulting files that are created adhere to the specified file sizes.

	5. Verify that the logs show a correct split size of 700 KB. 6. Further verify that the output files created are now less than the requested size. It is expected that after uploading a file that previously failed to split, the logs in Sedja show the correctly split size and the output files are of the right size specification.	
3	Manual Functional Test 1. Save a file that previously failed to split correctly as input to the updated version (e.g., a file of size 701,707 bytes with two separate pages was saved from the preliminary tests and before actualization) 2. Upload that file that previously failed to split and specify the same split size. (e.g., the previously failed split had a size of 700KB. This value is reused) 3. Verify that multiple files have been created and are all less than the requested size. (e.g., two separate files should be created from the original) It is expected that all files should be less than the requested size specifications.	Pass. The input file which previously failed named 1_PDFsam_Simba-GEAR1.pdf had an original size of 701,707 bytes after being split prior to the changes. Using this file and specifying the same split size of 700 KB, the output results in two separate files both under the specified limit.
4	Commit changes and perform code review in GitHub	Inspect changes from collaborative partner to ensure logical correction and understand

Time spent (in minutes): 45

9 Timing

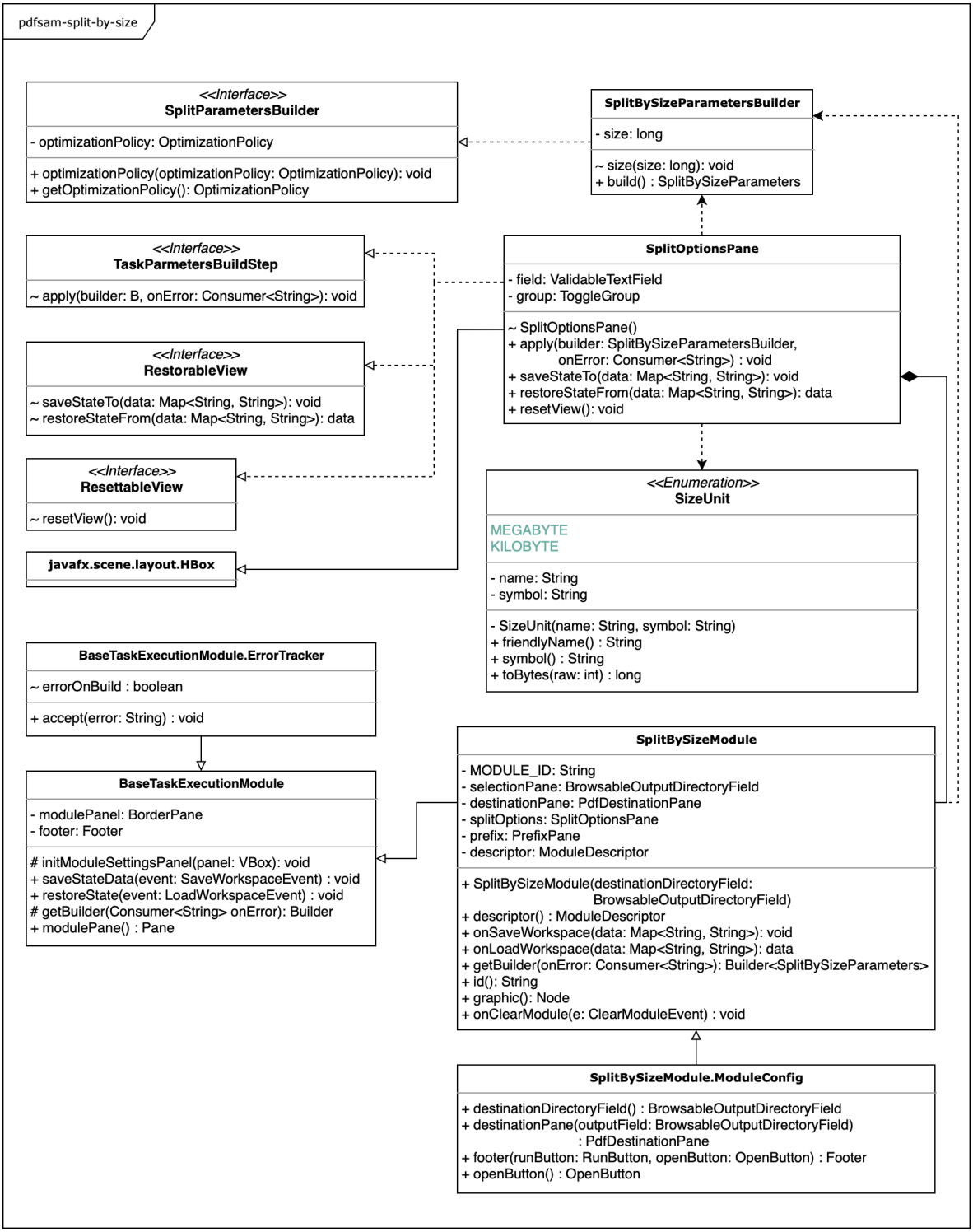
Summarize of the time spent in each step.

Phase Name	Time (in minutes)
Concept location	140
Impact Analysis	40
Prefactoring	0
Actualization	15
Postfactoring	0
Verification	45
Total	240

10 Reverse engineering

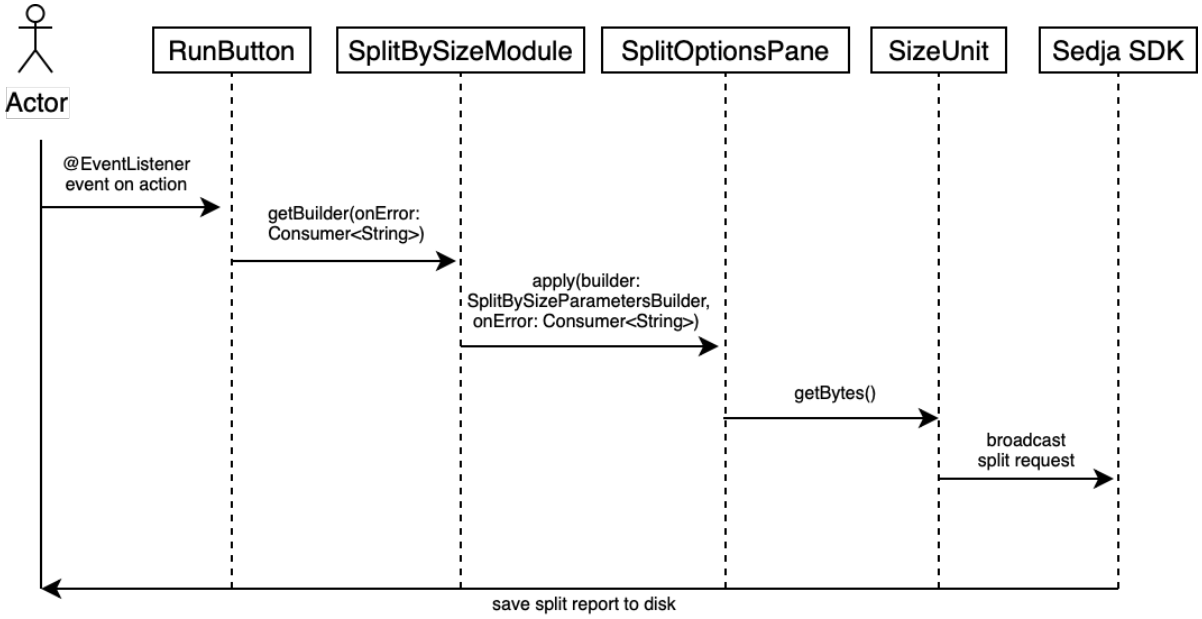
10.1 UML Class Diagram

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.).



10.2 UML Sequence Diagram

A UML sequence diagram corresponding to the main object interactions affected by the changes in this feature.



11 Conclusions

A challenging component for this feature request was the concept location and isolating a point of interest within the code. The primary reason for this was due to the attacking the problem with the intent to make a more significant modification. By increasing the default log level of the application to `DEBUG` and `TRACE` it was found that the actual PDF split was coordinated within the Sedja library. These files should not be modified to implement this feature request, so I thought that additional logic would have to be added after the run button was fired. Specifically, the initial assumption was that the violating files would have to read again from disk and be split again to ensure the files adhere to the feature standard.

After observing the logs further and understanding the sizes were producing logically correct results, I had to take a step back and trace the application from the start. This led to the latter section of concept location and the subsequent steps simply fell into place. Included with the change were several jUnit tests that had to be updated to reflect the change done during actualization. Additional manual functional tests were conducted as outlined in the validation section.

Classes and methods changed:

- `pdfsam-basic/src/main/resources/logback.xml`
    - updated log level for development purposes and should return to `INFO` for next release.
- `pdfsam-split-by-size/src/main/java/org/pdfsam/splitbysize/SizeUnit.java`
    - `@Override public long toBytes(int raw)` for both KILOBYTE and MEGABYTE enum values
- `pdfsam-split-by-size/src/test/java/org/pdfsam/splitbysize/SizeUnitTest.java`
    - `@Test public void toBytes()`
- `pdfsam-split-by-size/src/test/java/org/pdfsam/splitbysize/SplitOptionsPaneTest.java`
    - `@Test public void apply()`