Change request log

1 Team

Team Name: Swathy & Rachit

Swathy Hari Prasad : Change request log tracking

Rachit Dalal: Code inspection, Location identification of concept location and fixing the issue

2 Change Request

Change Request ID: #ps1

The Split by size module of PDFsam allows to split a PDF file in files of a given size (see Figure 5). Ideally, the created files should be smaller than the given size, but this does not always happen (see Figure 6). 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.

Code Location In Github:

parent location of our branch : Here you can see all the changes

https://github.com/SwathyAravind/cs515-001-s20-Swathy-Rachit-pdfsam/tree/dev-rachitswathy

Individual commits can be seen here

Commit- 1: To show the diff

https://github.com/SwathyAravind/cs515-001-s20-Swathy-Rachit-pdfsam/commit/a049a5002e21bdfdde560b01ff07212957564cec

Commit- 2: To show the diff

https://github.com/SwathyAravind/cs515-001-s20-Swathy-Rachit-pdfsam/commit/8ff394bb07d9ac6bb5b357142ae1670181550141

Commit- 3: To show the diff

https://github.com/SwathyAravind/cs515-001-s20-Swathy-Rachit-pdfsam/commit/15c8140d5aba66286e251e07535ed8c0f01f91bb

Commit- 4: To show the diff

https://github.com/SwathyIndividualAravind/cs515-001-s20-Swathy-Rachit-pdfsam/commit/b817dfd97a8a5ce63cfbb32078393e85325a7fbd

3 Concept Location

The following table below describes each step we followed when performing concept location for this change request. This table has various approaches such as IDE features that we used, search queries, class navigation, system workflow and execution, debugging, etc.

Step #	Description	Rationale

1	We ran the pdfsam split by size module to check the functionality and how it works. We performed different test cases to see how it works for MEGABYTES and KILOBYTES.	The rationale behind this is to get to know more about the working of the actual functionality and to get familiar with the functionality.
2	We put debug points in the code and also put some log statements to identify the exact location of the fix. Also we started application in debug mode.	To get the exact concept location.
3	After debugging we found org.pdfsam.splitbysize and org.pdfsam.core packages that can files for the concept location. 1> SizeUnit.java 2> SizeUnitRadio.java 3> SplitBySizeModule.java 4> SplitOptionsPane.java	We also then started identifying the probable location in the code base to and from those files where we can change the code to make it actual fix.
4	I also searched for "SplitBySizeParametersBuilder" in IDE and we got a total of 9 results which have this term. I opened SplitOptionsPane.java file under org.pdfsam.splitbysize package which has this keyword.	Rationale behind this is to identify all the places this has been used and from where it has been called and and how it has been called. We did this in the IDE searching tool by pressing "ctrl + shift + f" command with the exact match checked to be true.
5	I went to SplitOptionsPane.java because It is calling apply function and that function internally uses reference of SplitBySizeParametersBuilder class and builds the splitbysize module.	We did this to identify the workflow of the application and of the concept location.
6	With the SplitOptionsPane.java file we were not able to change anything so this was not the concept location.	To identify exact concept location for the fix.
7	Again I check for "kilobytes" and "megabytes" using IDE and I got 57 matches in total. I opened SizeUnit.java file	To identify all the places where it has been used. This time we did not search with match case and word checkbox to be true. So we got all the results.
8	In the SizeUnit.java file we found that conversion to bytes is happening and it is from ENUM type.	Therefore we reach the conclusion that something in this class needs to be changed.
9	Here the constant MEGABYTES and KILOBYTES are being multiplied by 1024 which we got to know that we have to change.	To identify exact location and the variables that need to be changed so that we were searching in SizeUnit.java class file
10	SizeUnit.java file and SizeUnit class is marked as a concept location.	This class has to be changed.

Time spent (in minutes): 350

4 Impact Analysis

The following table describes each step you follow when performing impact analysis for this change request. This contains the set of the classes that are related to the marked in the concept location and some of them which are

unchanged, changed or some may have side effects due to ripple effect depending on their impact which we have estimated.

Step #	Description	Rationale
1	We have looked in depth at SizeUnit.java to identify how splitting size module works. We got to know that it splits the pdf at 1024 which eventually exceeds the cut-off size.	As we have identified that SizeUnit.java file has the enumerated type of megabytes and kilobytes that converts to bytes. Therefore, we decided that this is the place where we can change something that will help us to make file size below the threshold.
2	Additionally, we found that there are no other classes that are directly related or dependent on the value of the enumerator that we have changed. We have not found any location in the code that relies on that value. This has been done using the tracing technique and what we have got as a result of this.	In order to verify that we have used how enumerated SizeUnit values have been used , where those values have been used and when those values have been used. After doing this we got to know how this can impact other places in the code.
3	SplitOptionsPane.java, SplitBySizeParametersBuilder.java, SplitBySizeModule.java, SizeUnitRadio.java Files have not changed.	While we were searching for concept location we identified that these files are interdependent but after analyzing them carefully we got to know that the change that we made in SizeUnit.java file is not creating any side effect on these files that's why we have not changed. To do this we have created a graph but it's on the rough paper so have not attached it here.
4	We have changed the conversion value from 1024 to 1000 to apply the fix.	Splitted file size should not exceed a given threshold size.
5	In total we have changed SizeUnit.java and SplitOptionsPaneTest.java and SizeUnitTest.java files.	These files have been located and these files will be impacted due to the change.

Time spent (in minutes): 98

5 Prefactoring (optional)- Not Done

Using the table below, describe each step you follow to refactor the code. Include as many details as possible, including the refactoring operations used (e.g., move method, extract class, etc.) and classes/methods/fields that 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

Time spent (in minutes): x

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.

Step #	Description	Rationale
1	The change we made in the SizeUnit.java file for conversion logic from MEGABYTES to KILOBYTES by adding this line. MEGABYTE = KILOBYTE.toBytes(raw) * 1000 KILOBYTE = raw * 1000	This fix will make sure that the size of splitted pdf should not exceed the given threshold compared to the original conversion which was there in conversion with these lines. MEGABYTE = KILOBYTE.toBytes(raw) * 1024 KILOBYTE = raw * 1024
2	Changed SizeUnitTest and SplitOptionsPaneTest classes.	In order to validate that the change in the code should not affect the application's behavior. Application should run as expected.
3	Finally the change has been committed to the Github to dev-rachitswathy branch.	In order to track those changes and for the future use.

Time spent (in minutes): 8

7 Postfactoring (optional) - Not Done

Use the table below to describe each step you followed to postfactor the code. Include as many details as possible, including the refactoring operations used (e.g., move method, extract class, etc.) and classes/methods/fields that 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

Time spent (in minutes): x

8 Validation

the below table describes validation activities (e.g., testing, code inspections, etc.) we performed for this change request. it includes the description of each test case, the result (pass/fail) and its rationale and after the actualization.

Step #	Description	Rationale
1	Test case defined: Manual Functional Test ● Build and run the mvn project. ● Open the Split by Size module from all the available options. ● Add a pdf file of valid size. let's say (10 MB) Split pdf at 1 MB. Expected output: Splitted pdfs size should not exceed the threshold size except for the single page.	The newly generated pdfs should not exceed the given split size. All the pdfs have size < threshold except a single page pdf whose size exceeds. Test result: Passed
2	Test case defined: Unit Test case Unit Test case in SizeUnitTest.java. ● We changed the unit test case to conversion constant from 1024 to 1000 in toBytes() method's assert statement to ensure that file should not exceed the size of the given threshold and existing unit test should pass. Expected output: It should pass the test case with the given threshold during the conversion to maintain the functionality.	All the test cases related to this should be passed after the change in the sizeUnit.java and corresponding changes into SizeUnitTest.java file. The test passed.
3	Test case defined: Unit Test cases in SplitOptionsPaneTest.java. ● We changed the unit test case to conversion constant from 1024 to 1000 into the **public void apply()** method to ensure that file should not exceed the size	The test cases related to this should be passed while performing apply function which gets called on the footer.runButton click. SplitOptionsPaneTest.java file hase apply function which does this functionality of splitting the pdf and therefore all the test cases should be passed.

	of the given threshold for the apply function and existing unit test should pass. Expected output: It should pass the test case with the given threshold during the conversion while running the apply function.	The test passed.
4	Test case defined: Manual Functional Test ● Keep the file that is not correctly split to the given size. Provide those files as a input to the updated version (For example: file size 505, 509, 514 KB with 3 separate pdfs was kept before the actual changes implemented) ● Upload those files which are failed in the before the actualization and after actualization specify the same split size and see the output. (the failed files had a split size of 500 KB. This split size will be used again. ● Validate that many files have been created and all of them are below threshold. Expected Output: All the files should be less than the given threshold values.	All the files which were provided as an input had size exceeded then the given split size(here split size was 500 KB). New generated files should be less than the specified split size has been observed. The test passed.

Time spent (in minutes): 51

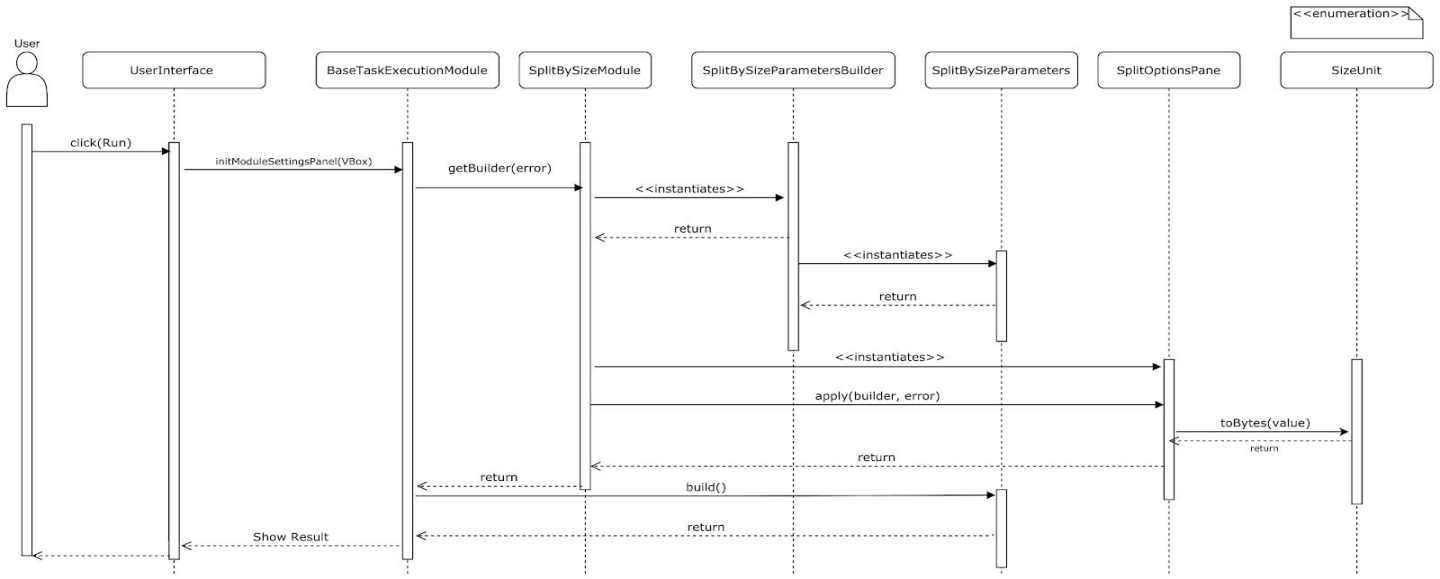
9 Timing

Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	450
Impact Analysis	98
Prefactoring	0
Actualization	8
Postfactoring	0
Verification	51
Total	607

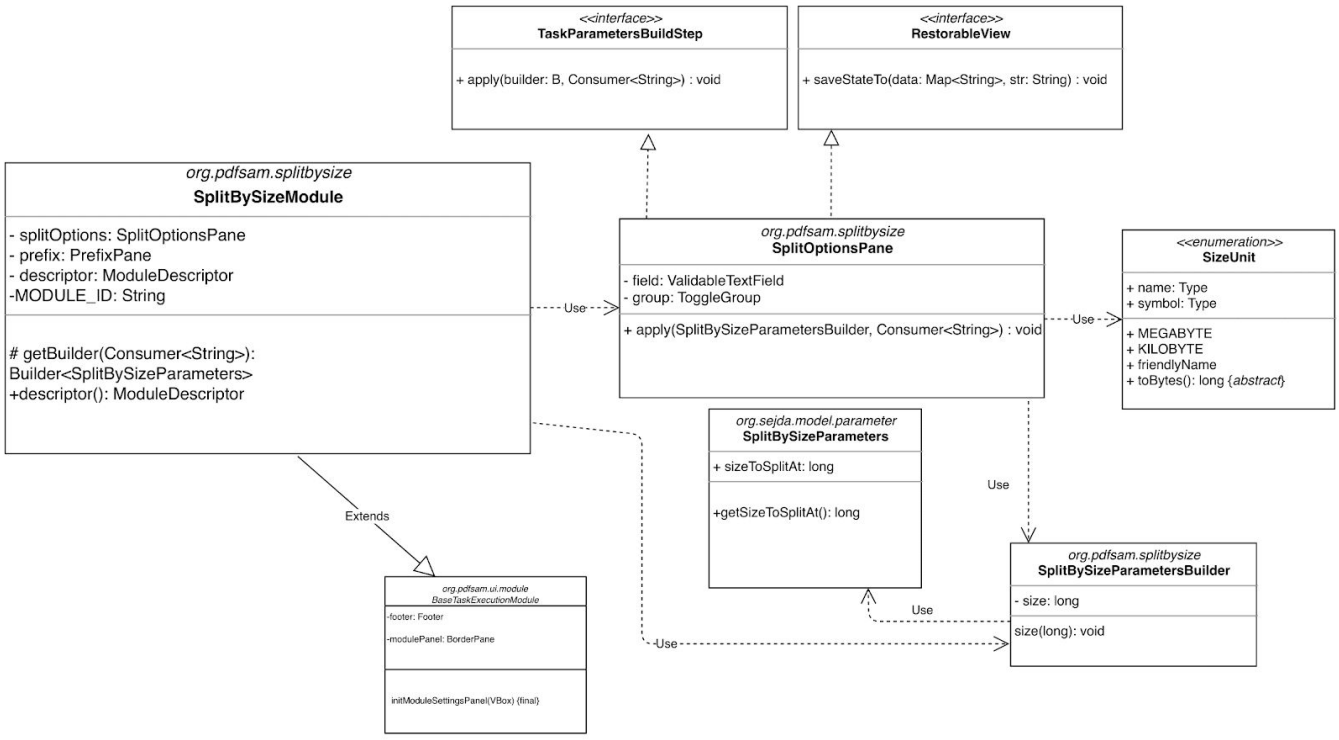
10 Reverse engineering

A UML sequence diagram corresponding to the main object interactions affected by our changes.



UML Class Diagram

A UML class diagram of the classes which are visited during the task of identifying concept location. This includes the different types of relationship between the classes. (aggregation, inheritance, composition, dependency, etc..)



11 Conclusions

This change request would have been easy if we would have identified the change location in the early stage. After doing a lot of debugging and log statements we got to know that abstract method toBytes() does the magic. By saying this we meant that the class SizeUnit.java inside which the conversion from raw input to KILOBYTES and MEGABYTES happens is the place that needs to change in order to split the pdf with the given threshold value.. After doing a lot of testing we narrowed down the location was inside org.pdfsam.splitbysize package and specifically SizeUnit.java file and we changed value of 1024 to 1000 and confirmed the same by doing manual testing and also the unit test cases of the same have also changed.

Classes and methods changed:

- pdfsam-split-by-size/src/main/java/org/pdfsam/splitbysize/SizeUnit.java
    - both ENUM values KILOBYTE & MEGABYTE @Override public long toBytes(int raw) method has been changed

- pdfsam-split-by-size/src/test/java/org/pdfsam/splitbysize/SizeUnitTest.java
    - @Test public void toBytes() -> assertEquals statement changed
- UNIT TEST
    - Original Statement - assertEquals(5 * 1024, SizeUnit.KILOBYTE.toBytes(5));
    - Modified Statement - assertEquals(5 * 1000, SizeUnit.KILOBYTE.toBytes(5));

    - Original Statement - assertEquals(5 * 1024 * 1024, SizeUnit.MEGABYTE.toBytes(5));

    - Modified Statement - assertEquals(5 * 1000 * 1000, SizeUnit.MEGABYTE.toBytes(5));

- pdfsam-split-by-size/src/test/java/org/pdfsam/splitbysize/SplitOptionsPaneTest.java
- UNIT TEST
    - @Test public void apply() -> verify statement changed
    - Original Statement - verify(builder).size(eq(30 * 1000 * 1024L));
    - Modified Statement - verify(builder).size(eq(30 * 1000 * 1000L));