Change request log

1 Team - bughunters

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

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#ps1)

2 Change Request

Change Request Id – ps1

Description - 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.

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 ps1:

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 input with different input files (pdf files containing high res images or just text or combination of both). ● Observe the output files and their size. ● There is only one case when the output file size is significantly bigger than the expected size. This is in the case of pdf document having high res images. Otherwise the difference Is not significant.	To get an idea of what difference in size occurs in actual output files compared to expected file size.
3	● Observe the dependency graph to analyze the modules that might be involved in the complete split by size workflow. ● IDE Feature used: Dependency Graph for modules. ● Possible modules: ➢ pdfsam-basic ➢ pdfsam-core ➢ pdfsam-service ➢ pdfsam-gui/pdfsam-fx ➢ pdfsam-split-by-size (identified as the most	Try to identify modules involved by studying the software design.

	important module based on the knowledge of 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. ➢ It's the pdfsam-basic module, which is called to start the application. ● There is a 'Run' button in the UI module 'Split by size', 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 **'SplitBySizeModule' class** (**line 189**). ➢ 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.	Some code navigation as preparation for debugging the application.
5	Debug the application to identify all the components involved in the split module.	Try to understand the workflow implemented for the split module.
6	The debug process was run several times to narrow down the part/file responsible for setting the size of the output files as well splitting the files based on the size. 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.	Try to locate the code which splits the source file in parts.
7	Debug the application and step into the sejda SDK implementation to check if there is any issues while splitting the file related to size of output files.	Try to understand how sejda SDK splits the file.
8	● We found that the application performed all the steps correctly as the size of output files in bytes was correct. ● This lead to checking of output file size in bytes, and we found that the size in bytes is still under the specified size. ● The size displayed by OS(s) is in 'kilo' and 'mega' units, whereas the size calculated in pdfsam-split-by-size module was in 'kibi' (line 32 and line 38 in SizeUnit class in pdfsasm-split-by-size module).	Try to check the output file size again and identify root-cause.

	● The difference in 'kilo' and 'kibi' is the number of bytes considered. For **'kilo' it is 1000**, and **1024 for 'kibi'**. If size is in 'kibi' units then it is denoted in 'kiB' whereas 'kB' is used for 'kilo'.	
9	We identified the class SizeUnit as the root-cause.	We confirmed this class had to be modified.

Time spent (in minutes): 150

4 Impact Analysis

The impact analysis for this change was not very complicated as the change is related to how the size of any file is calculated by the software as well as OS.

Step #	Description	Rationale
1	● We tried to find the usages of SizeUnit class in all modules by using the 'Find usage' functionality of the IDE. ● We found no classes having direct impact due to identified change. ● We did track a hierarchy of classes propagating this class, they are as follows: SizeUnit → SizeUnitRadio → SplitOptionsPane → SplitBySizeModule	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. ● The only other class to be changed is the 'SizeUnitTest' which contains the unit tests for the class 'SizeUnit'.	Track all the classes to be changed.

Time spent (in minutes): 30

5 Prefactoring (optional)

The only change needed is to change the constants to calculate the raw number of bytes for each output file. There is no need for prefactoring needed for this change request.

Time spent (in minutes): 0

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	We changed the constants 1024 to 1000 in 'SizeUnit' class, in two methods (line 32 and 38). These methods were for getting size in bytes for 'Kilobyte' and 'Megabyte' option.	The size should be in kilo and not in kibi.

2	We changed the expected output for the unit tests.	The criteria for unit testing also changes.

Time spent (in minutes): 20

7 Postfactoring (optional)

There is no need for postfactoring as the change is just in a constant field.

Time spent (in minutes): 0

8 Validation

We ran unit tests as well as manual tests for confirming correct behavior as well as output.

Step #	Description	Rationale
1	Unit Test Test case defined: Input size in Kilobytes and Megabytes should confirm with total number of bytes Inputs: 5 (for Kilobytes), 5 (for MegaByte) Expected output:5000, 5000000	This is the regular expected behavior. The test passed.
2	Manual Test Test: Normal pdf file should be split in output file of size less than or equal to specified split size Input: Input file and split size:10 MB Expected Output: Output files with size less than 10MB	This is the regular expected behavior. The test passed.

Time spent (in minutes): 15

9 Timing

Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	150
Impact Analysis	30
Prefactoring	0
Actualization	20
Postfactoring	0
Verification	15
Total	215

10 Reverse engineering

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

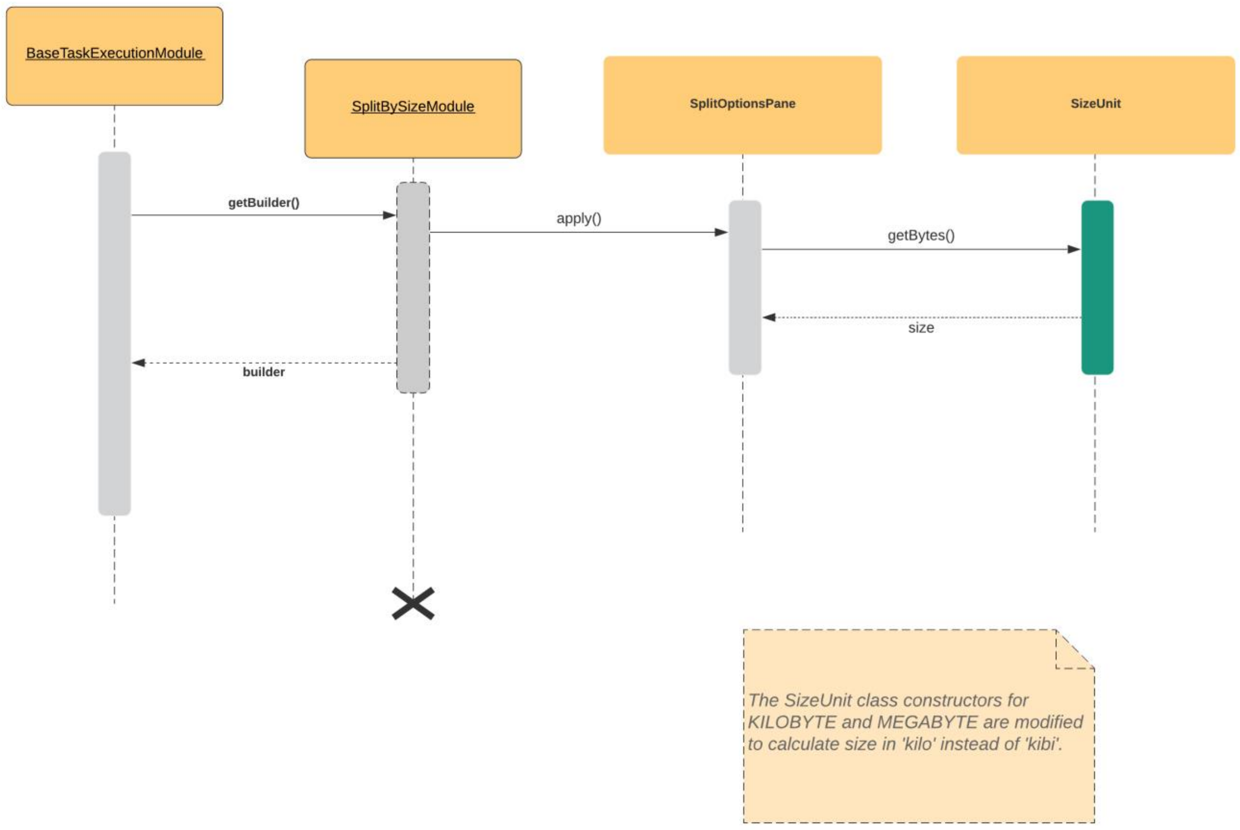
BaseTaskExecutionModule
SplitBySizeModule
SplitOptionsPane
SizeUnit
getBuilder()
apply()
getBytes()
size
builder
The SizeUnit class constructors for KILOBYTE and MEGABYTE are modified to calculate size in 'kilo' instead of 'kibi'.

Diagram 1: UML Sequence diagram

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

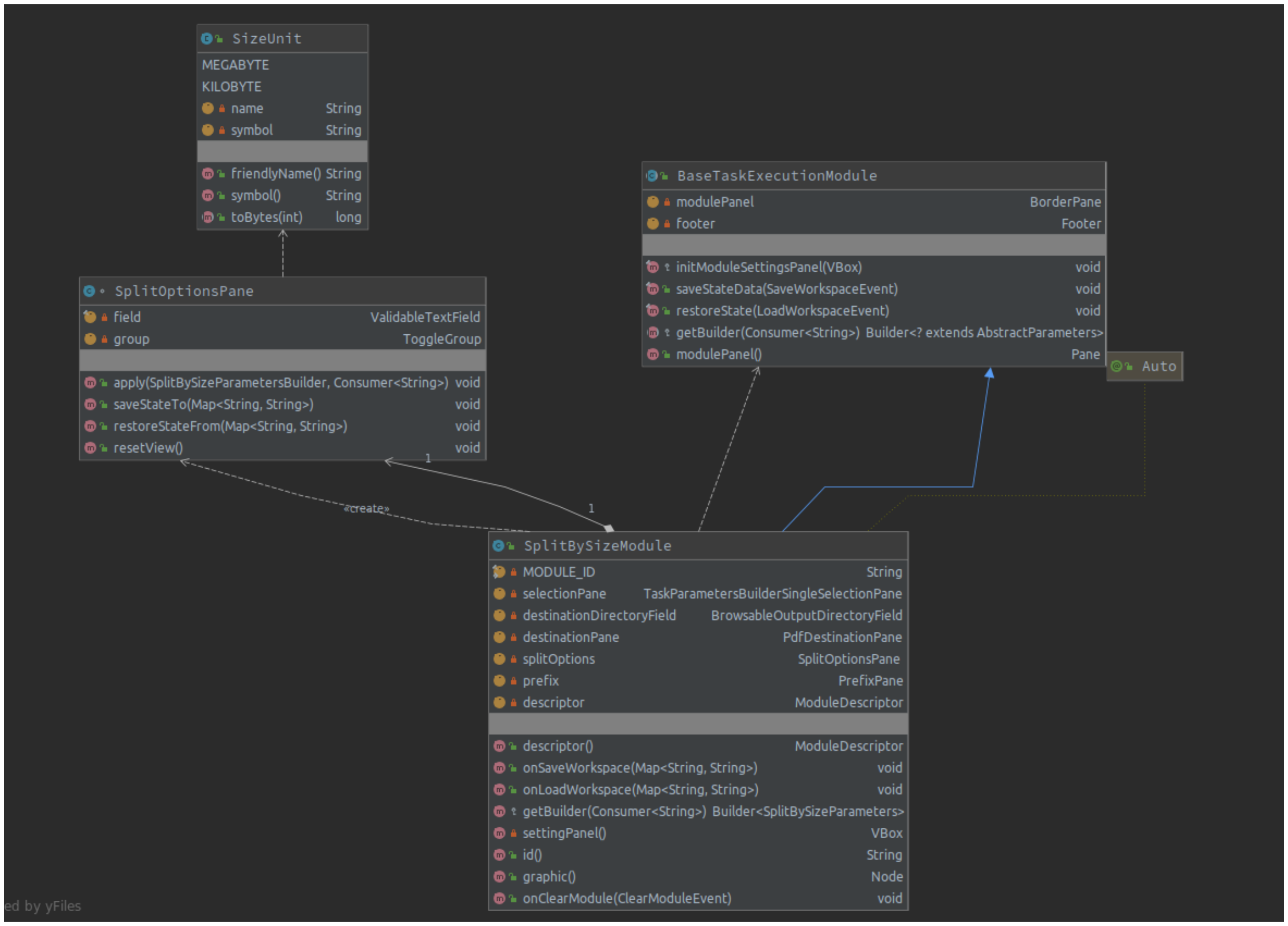
SizeUnit
MEGABYTE
KILOBYTE
name String
symbol String
friendlyName() String
symbol() String
toBytes(int) long
BaseTaskExecutionModule
modulePanel BorderPane
footer Footer
initModuleSettingsPanel(VBox) void
saveStateData(SaveWorkspaceEvent) void
restoreState(LoadWorkspaceEvent) void
getBuilder(Consumer<String>) Builder<? extends AbstractParameters>
modulePanel() Pane
Auto
SplitOptionsPane
field ValidableTextField
group ToggleGroup
apply(SplitBySizeParametersBuilder, Consumer<String>) void
saveStateTo(Map<String, String>) void
restoreStateFrom(Map<String, String>) void
resetView() void
1
«create»
1
SplitBySizeModule
MODULE_ID String
selectionPane TaskParametersBuilderSingleSelectionPane
destinationDirectoryField BrowsableOutputDirectoryField
destinationPane PdfDestinationPane
splitOptions SplitOptionsPane
prefix PrefixPane
descriptor ModuleDescriptor
descriptor() ModuleDescriptor
onSaveWorkspace(Map<String, String>) void
onLoadWorkspace(Map<String, String>) void
getBuilder(Consumer<String>) Builder<SplitBySizeParameters>
settingPanel() VBox
id() String
graphic() Node
onClearModule(ClearModuleEvent) void
ed by yFiles

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 the most challenging step. Once concept location was completed rest of the process went by smoothly. The understanding of the software design helped immensely in locating the root-cause. We used IntelliJ IDEA IDE for all the steps performed (implementation and testing).

Classes and methods changed:

- /pdfsam-split-by-size/src/main/java/org/pdfsam/splitbysize/SizeUnit.java
    - MEGABYTE (constructor)
    - KILOBYTE (constructor)

- /pdfsam-split-by-size/src/test/java/org/pdfsam/splitbysize/SizeUnitTest.java
    - toBytes()