Change request log

# Team

Team: Team356

Members : Aaron Pereira and Ryan Becwar

# Change Request

Change Request #2 (PDFSam)

The Alternate Mix module allows to merge two or more documents taking pages alternately in natural or reverse order. Of course, this feature does not work when only one document is provided (see Figure 6). You are requested to modify this module to allow the reversal of a single document.

# Concept Location

|  |  |  |
| --- | --- | --- |
| **Step #** | **Description** | **Rationale** |
| **1** | We ran the system. Put in just a single input file. Clicked on ‘Reverse’ and observed what happened. |  |
| **2** | The output in eclipse gave us a fancy stack trace. | Since we had an exception that was being thrown (With a suitable error message), We were initially inclined to follow the stack trace to get a fair idea of what was wrong. |
| **3** | We Visited the following class and observed what was happening in the Validate() method.  org.sejda.core.service.DefaultTaskExecutionService.validate(DefaultTaskExecutionService.java:116) | Finding Out what was happening here would probably lead us to a possible root of the validation logic |
| **4** | We visited at least 30 different classes placing breakpoints along the way without much luck. At this point we were debugging Sejda library code( And also hibernate code). |  |
| **5** | Stepping into ValidatorImpl.validate() lef us to understand that pdfsam reads ‘Constraints’ from somewhere. We further stepped intoValidatorImpl.validateInContext() to see what was happening. We realized that here ‘BeanMetaData ’ was being picked up from the current bean. Following the execution by stepping into successive functions. | We needed to understand where these constraints originate from. |
| **6** | We finally found an ‘@AtLeastTwo’ annotation defined in ‘AlternateMixMultipleInputParameters’. This annotation was being read by Sejda’s Alternate Mix and was causing the validation exception | We needed to understand where these constraints originate from. |
| **7** | Changes could not be made to sejda. So we tried a variety of tricks to ignore/change/skip this annotation. None of them seemed to work. We then concluded that the bean ‘AlternateMixMultipleinputParameters’ had to be changed in ‘AlternateMixParametersBuilder’ in order to skip this annotation. | This annotation had to skipped for this requirement to be satisfied |
| **8** | Class to change was finalized as ‘AlternateMixParametersBuilder’ |  |

**Time spent (in minutes):** 420

# Impact Analysis

|  |  |  |
| --- | --- | --- |
| **Step #** | **Description** | **Rationale** |
| **1** | We created our own new bean class that extended AbstractPdfOutputParameters and did not place the ‘AtLeastTwo’ annotation here | We rationalized that the only way to not have the annotation imported was to make our own bean class without the annotation |
| **2** | We inspected where else the original ‘AlternateMixMultipleinputParameters’ being used. A file search led us to mark ‘AlternateMixParametersBuilder’ and ‘sejda.xml’ as the only files that were directly affected by our change | We realized that a ‘file’ search would be the easiest and best way to zero down on all the files that could be impacted by our change. We just had to figure out where this class was being instantiated/referenced or extended. |

**Time spent (in minutes):** 60

# Prefactoring (optional)

**We did not follow this step.**

# Actualization

|  |  |  |
| --- | --- | --- |
| **Step #** | **Description** | **Rationale** |
| **1** | We created two classes ‘AlternateMixInputParameters’ and ‘AlternateMixSingleTask’. ‘AlternateMixInputParameters’ had to be made without the @AtLeastTwo annotation and ‘AlternateMixSingleTask’ would now use ‘AlternateMixInputParameters’ internally. | We realized that the only way to bypass the @AtLeastTwo annotation was to define our own bean class. ‘Sejda.XML’ also showed us that we needed to redefine a ‘Task’ for this bean so we created a task too. |
| **2** | Sejda.XML was modified to include our newly created task | PDFsam dynamically loads a bunch of tasks from this XML file at startup using Reflection. During the Constraint extraction and ‘Bean instantiation’ process. Task Metadata had to be added for our new task in Sejda.XML |
| **3** | AlternateMixParametersBuilder.java had to be modified to now extend ‘SingleOutputTaskParameterBuilder’ that took a generic of our newly created ‘AlternateMixInputParameters’. It also had a bunch of internal instantiation of the same class that had to be modified too | This had to be done to get our CR to work |
| **4** | An in-built test case for this module had to be changed to use ‘AlternateMixInputParameters’ too. | To ensure Testing was done correctly. |

**Time spent (in minutes):** 25

# Postfactoring (optional)

**We skipped this step.**

# Validation

|  |  |  |
| --- | --- | --- |
| **Step #** | **Description** | **Rationale** |
| **1** | Test case defined: UI testing the Alternate Mix module for expected behavior after implementing the CR.  Inputs: upload a single file in the module and click on reverse. Click on run.  Expected output: The same input document in reverse order. | Expected behavior after implementing the CR. The test passed. |
| **2** | Test case defined: UI testing the Alternate Mix module for expected behavior irrespective of implementing the CR.  Inputs: upload 3 Documents and select reverse in any 1. Click on run.  Expected output: the 3 input PDFs alternately mixed with one document in reverse. | This is the regular expected behavior.  The test passed. |
| **3** | Test case defined: UI testing the Alternate Mix module for expected behavior irrespective of implementing the CR.  Inputs: Do not upload any document  Expected output: validation error | This is the regular expected validation behavior. The main goal was to test if the other validations still work.  The test passed. |
| **4** | JUnit test cases.  Running the inbuilt Junit test cases for all existing cases.  Inputs : Both Pre-defined Junit test cases for Alternate Mix.  Expected Output : No Junit errors | These tests should pass to ensure that existing functionality of the same module was not affected by the change.  Both Tests passed. |

**Time spent (in minutes):** 60

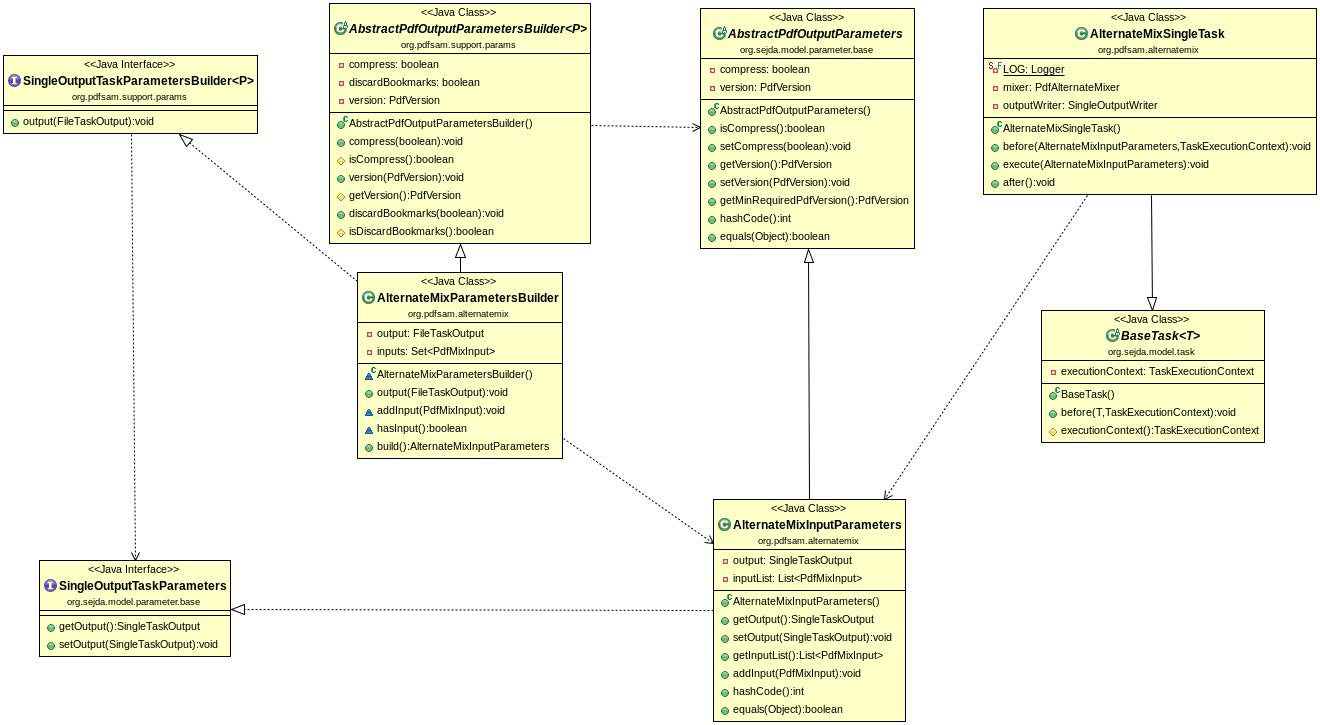
# Timing

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| **Phase Name** | **Time (in minutes)** |
| Concept location | 420 |
| Impact Analysis | 60 |
| Prefactoring | - |
| Actualization | 25 |
| Postfactoring | - |
| Verification | 60 |
| **Total** | 565 |

# Reverse engineering

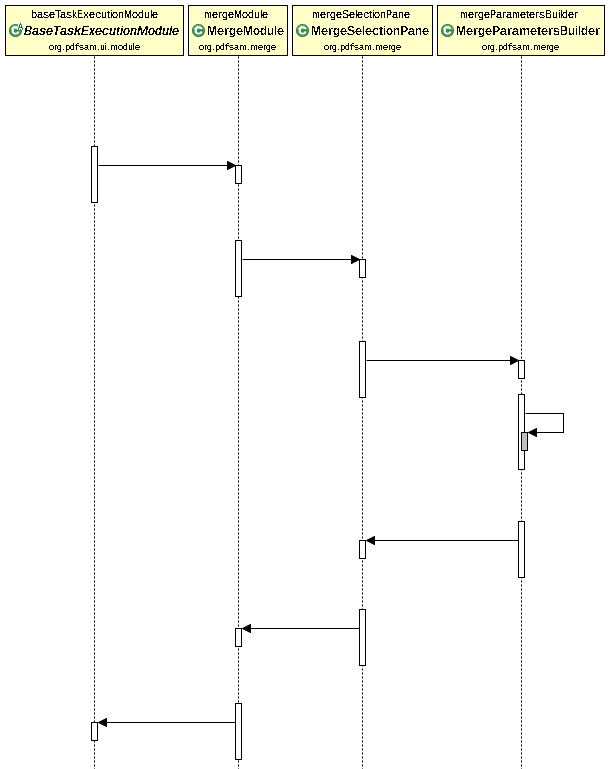
**UML Class Diagram**



**Sequence Diagram:**

In this sequence Diagram, the trace of how the AddInputs Method is reached is described. The loop in MergeParametersBuilder signifies the execution of our code change that modifies page ranges.

**Sequence Diagram**



# Conclusions

This Change Request was very challenging because the original author made changes in the Sejda module and just used said changes in PDFsam. If we could have modified Sejda, this CR would only involve getting rid of the @AtLeastOne annotation (One line change). Instead we had to think of a work around that involved creating a bunch of our own classes and then using them that needed changes in multiple files. Repeated used of Sejda and Hibernate to do input validation and constraint checking also made debugging very lengthy and difficult that took up most of our time. We also have to admit that this approach could pose some serious maintainability questions. It also has a lot of replicated code. If the classes in Sejda change, we would also have to modify our classes. With the given restrictions with modifying dependencies, this is the best approach we could come up with that fully satisfied the requirement. If these restrictions were not in place, this CR would have been extremely easy to implement.

Classes and methods Added:

* org.pdfsam.alternatemix.AlternateMixInputParameters
* org.pdfsam.alternatemix.AlternateMixSingleTask

Classes, methods and metadata files changed:

* org.pdfsam.alternatemix.AlternateMixParameterBuilder
  + Class definition
  + build()
* org.pdfsam.alternatemix.AlternateMixParametersBuilderTest
  + build()
* pdfsam-basic/src/main/resources/sejda.xml