Change request log

# Team

Team reNo

Matt Dragan and Ben Sattelberg

# Change Request

Change request 3.2

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

Use the table below to describe each step you follow when performing concept location for this change request. In your description, include the following information when appropriate:

* IDE Features used (e.g., searching tool, dependency navigator, debugging, etc.)
* Queries used when searching
* System executions and input to the system
* Interactions with the system (e.g., pages visited)
* Classes visited
* The first class found to be changed (this is when concept location ends)

When there is a major decision/step in the process, include its rationale, i.e., why that decision/step was taken.

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

The following is an example of a concept location process for the change request "Color student schedule":

|  |  |  |
| --- | --- | --- |
| **Step #** | **Description** | **Rationale** |
| **1** | *We ran the system* |  |
| **2** | *We interacted with the system: we ran a few variations of pdfs through the alternate mix functionality to determine how it worked.* | *To acquire familiarity with the system and understand the functionality of the module.* |
| **3** | *We noticed that when two pdfs are given but one has a page range larger than the number of pages in the document, it is not used in the output.* | *This suggests at least one way this problem could be fixed, although it is likely not a good approach.* |
| **4** | *We looked through the pdfsam-alternate-mix module.* | *This module will likely contain the location of what is to be modified.* |
| **5** | *We looked through the pdfsam-basic, pdfsam-core, and pdfsam-fx modules.* | *This is a long step. We were trying to determine where in pdfsam functionality actually occurred. We found that each of these modules had useful functionality, but none of them actually modified pdfs. We did realize most files imported pieces of the “sejda” package.*  *This step consisted of many substeps, but we were getting unfortunately disheartened during this portion of concept location and did not keep perfect track of what exact steps we took.* |
| **6** | *We investigated pieces of the sejda module through Eclipse’s ability to jump to declarations.* | *We wanted to determine what sejda was doing. We did encounter difficulty here where Eclipse would frequently stop allowing us to go to the declarations of files. At a certain point, we stopped being able to entirely.* |
| **7** | *We searched for “pdfsam sejda” online to both investigate what sejda is and how pdfsam uses it (and in hopes of finding a repository we could look at when we weren’t able to view the sejda code through Eclipse).* | *Since we were having difficulty understanding what sejda was, we wanted to investigate any resources online.* |
| **8** | *We read the blogpost at* [*https://pdfsam.org/new-release/busy-with-sejda-v2/1425/*](https://pdfsam.org/new-release/busy-with-sejda-v2/1425/) | *We were still trying to figure out what relation sejda had with pdfsam. We determined that sejda was the base structure used for modifying pdfs that pdfsam used the API for.* |
| **9** | *We read through the sejda website* | *We wanted to determine the functionality of sejda. From this, we determined that we likely wouldn’t actually be changing how pdfs are interacted with in our changes, merely the way in which sejda commands were called.* |
| **10** | *We revisited the alternate-mix module.* | *We wanted to determine how pdfsam was interacting with the sejda structures.* |
| **11** | *We revisited the pdfsam-basic, pdfsam-core, and pdfsam-fx modules.* | *We weren’t able to determine what was going on in the alternate-mix module, so we looked towards structures that were hopefully less complex/specific.* |
| **12** | *We looked at the pdfsam-rotate module.* | *Advice was given in class to investigate this module, as it was the simplest to understand. It helped us a great deal in figuring out what was going on, so we felt hopeful in revisiting the alternate-mix module.* |
| **13** | *We looked at the “AlternateMixModule.java” class* | *This seemed to be the most likely class to implement functionality. We found that it seemed to construct the other classes in this module and structure portions of the display. We didn’t find it likely that this would be something we needed to change.* |
| **14** | *We looked at the “AlternateMixParametersBuilder.java” class* | *This seemed like the second most likely class to implement functionality. We found that it mostly overrode functions from classes it inherited or implemented. Based on the structure of this, we found it unlikely that it would be what we needed to change. We did note that it extended AbstractPdfOutputParametersBuilder with a type of AlternateMixMultipleInputParameters.* |
| **15** | *We looked for the AlternateMixMultipleInputParameters.java class* | *We wanted to determine if there was an AlternateMixSingleInputParameters class, or if there would be a way to use only one input. We found that this was a sejda class (so not something we could change) and that it strongly required at least two input files.* |
| **16** | *We looked at the “AlternateMixSelectionPane.java” class* | *We moved to the remaining file in the alternate-mix module. We found that it seemed to construct and store the table of inputs and their options. We also found the apply function, which seemed to us to implement the functionality of what we were looking for.* |
| **17** | *We tentatively marked “AlternateMixSelectionPane.java” as located.* | *This seemed like the most likely class to be modified.* |

**Time spent (in minutes):** ~390

# Impact Analysis

Use the table below to describe each step you follow when performing impact analysis for this change request. Include as many details as possible, including why classes are visited or why they are discarded from the estimated impact set.

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

Do not take the impact analysis of your changes lightly. Remember that any small change in the code could lead to large changes in the behavior of the system. Follow the impact analysis process covered in the class. Describe in detail how you followed this process in the change request log. Provide details on how and why you finished the impact analysis process.

|  |  |  |
| --- | --- | --- |
| **Step #** | **Description** | **Rationale** |
| **1** | *We made a list of methods called by apply in AlternateMixSelectionPane.java* | *To track the classes that could be impacted by the change.* |
| **2** | *We inspected the class AlternateMixParametersBuilder. It was not marked for changes.* | *Although this is the object that is being used by apply, there don’t seem to be any changes we are able to make that don’t require modifying AlternateMixMultipleInputParameters.java in the sejda package as well.* |
| **3** | *We inspected the class “ConversionUtils.java.” It was marked to change.* | *The toPageRangeSet(String) function will be useful, but it will be more useful to have a toPageRangeSet(int) function.* |
| **4** | *We inspected the class “SelectionTableRowData.java.” It was not marked for change.* | *Although this class uses ConversionUtils and is used by AlternateMixSelectionPane when building the builder, it does not have functionality that needs to be changed.* |
| **5** | *We made a list of classes that may call AlternateMixSelectionPane.java’s implementation of apply.* | *To track the classes that could be impacted by the change. However, it does not appear that any classes call this function in pdfsam. Tracking what may indirectly call it through sejda is unknown and should not matter for these changes.* |

**Time spent (in minutes):** 43

# Prefactoring (optional)

Using the table below, describe each step you follow to prefactor 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** |
| **1** | *We added a method in ConversionUtils.java that has equivalent functionality as the original toPageRangeSet(String selection), but accepts an integer parameter instead (it in fact calls the original function so as not to reuse code). We are unsure what refactoring technique this is and were not able to find a name for it.* | *We need to create a Set<PageRange>, but we will be using an integer number of pages rather than a String entered by the user. Adding this functionality is more explicit and prevents the selection pane from having to parse data.* |
| **2** | *After the previous change, we ran the unit tests and we ran the system.* | *We tested everything was working as before, after the refactoring. This method was not connected to anything, so success was expected.* |
| **3** | *We committed our changes with git.* | *Just in case we need to revert our changes.* |

**Time spent (in minutes):** 7

# 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** | *In AlternateMixSelectionPane’s apply method, we added a case for if the length of the table is 1.* | *We need different logic when there is only one element in the table.* |
| **2** | *In the new case, we use the original logic for adding the existing table element to the builder, but we additionally add a table element that will result in an empty page range.* | *This is likely not the best implementation of this. However, the AlternateMixInputParameters in sejda requires multiple inputs. To get around this, we add another input that is the original pdf with a page range greater than the maximum number of pages in the pdf.* |
| **3** | *We ran the existing test cases.* | *To make sure everything works. We found that there was a test case related to ensuring if there was one element in the table that the builder did not go through with execution, so we removed this test case.* |
| **4** | *We did functional testing of the system to ensure that the expected behavior occurred.* | *Single pdfs were reversed as expected.* |

**Time spent (in minutes):** 23

# Postfactoring (optional)

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** |
| **1** | *We did not find any postfactoring necessary.* | *The changes made were relatively isolated. We considered extracting a method for adding the input to the builder, but we did not feel it would fit with the overall architecture of the system and would not contribute significantly to the flow of the code. Discussion of this accounts for the time spent in this step.* |

**Time spent (in minutes):** 5

# Validation

Use the table below to describe any validation activity (e.g., testing, code inspections, etc.) you performed for this change request. Include the description of each test case, the result (pass/fail) and its rationale.

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

|  |  |  |
| --- | --- | --- |
| **Step #** | **Description** | **Rationale** |
| **1** | *Code review between the members of our group.* | *We wanted to ensure that the changes we made worked and matched the change request.* |
| **2** | *Functional testing: we input a single pdf and ensured that it reversed successfully with no extra pages.* | *This is the intended behavior of the change.* |
| **3** | *Functional testing: we input multiple pdfs and verified they were mixed correctly.* | *This was the original behavior of the system. We wanted to ensure it continued to work.* |
| **4** | *Ideally we would add at least one unit test corresponding to reversing a pdf.* | *Providing automated testing of this would be useful for ensuring validity. However, the structure of the existing unit tests was not something we were able to understand to replicate. Attempting to understand them accounts for the bulk of time spent in this step.* |

**Time spent (in minutes):** 72

# Timing

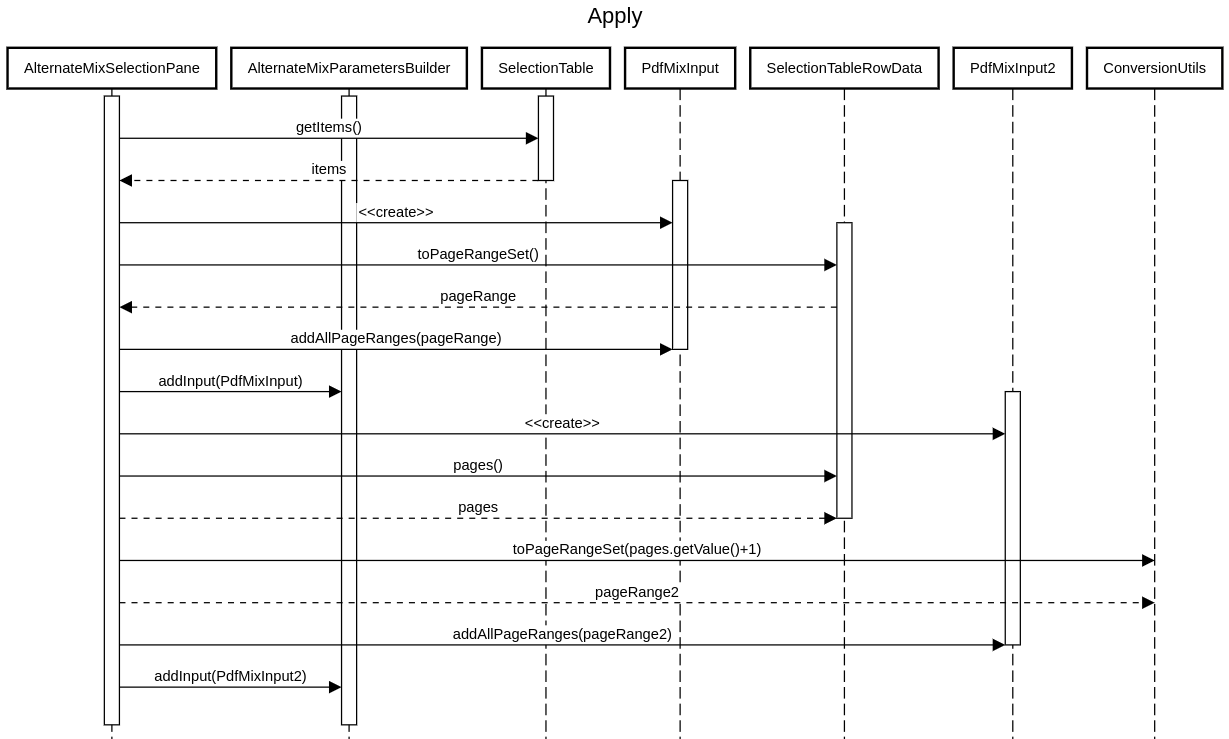
Summarize the time spent on each phase.

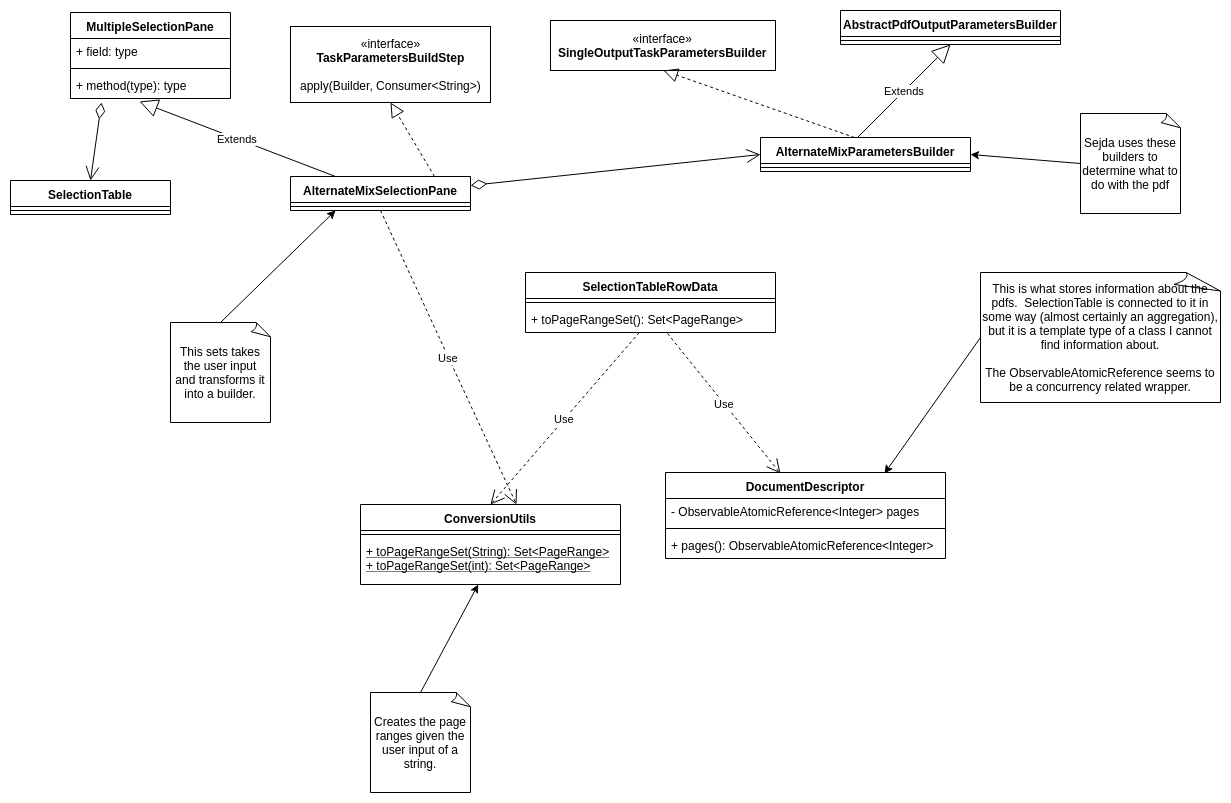
|  |  |
| --- | --- |
| **Phase Name** | **Time (in minutes)** |
| Concept location | ~390 |
| Impact Analysis | 43 |
| Prefactoring | 7 |
| Actualization | 23 |
| Postfactoring | 5 |
| Verification | 72 |
| **Total** | ~540 |

# Reverse engineering

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

Create a partial UML class diagram of the classes visited while navigating through the code. Include the associations between classes (e.g., inheritance, aggregations, compositions, etc.), as well as the important fields and methods of each class that you learn about. The diagram may have disconnected components. Use the UML tool of your preference. When a significant fact about a class or method is learned, indicate it via annotations on the diagram. **For each change request, start with the diagram produced in the previous change request. For the first, you will start from scratch.**

****

****

# Conclusions

Provide a set of conclusions about the change request and the change process. List the major challenges this change request posed.

List all the classes and methods you have changed.

For example:

*For this change, concept location was a significant challenge. Neither of us is particularly experienced with Java and most of the systems we interact with do not utilize significant aspects of OOP, so we needed quite a bit of time to understand what was going on. Many of the patterns that were followed in designing pdfsam were novel to us. Figuring out the divide between pdfsam and sejda was also a challenge, as the inheritance and implementation of an exterior library’s structures was difficult to follow. We also suspect that our method for implementing this change request is not a strong method of doing it, but we were unsure of how it could be done better. Concept location was done using Eclipse’s search functionality, and the user documentation for sejda and pdfsam. Impact analysis was done using Eclipse’s search functionality. Automated validation was difficult, as we were not able to fully understand the unit tests so we could not add additional ones to test our added functionality. The UML class diagram was created with draw.io. The sequence diagram was created on https://sequencediagram.org/*

*Ben implemented these changes with input from Matt (note that the prefactoring commit was sent on Matt’s account due to issues with connecting to github on Ben’s machine at that time). Ben wrote the documentation and created the UML diagrams.*

*Classes and methods changed:*

* *pdfsam-parent/pdfsam-alternate-mix/src/main/java/org/pdfsam/alternatemix/AlternateMixSelectionPane*
  + *void apply(AlternateMixParametersBuilder builder, Consumer<String> onError)*
* *pdfsam-core/src/main/java/org/pdfsam/support/params/ConversionUtils*
  + *Set<PageRange> toPageRangeSet(int selection)*