**CS/SE 6356 Software Maintenance, Evolution & Re-engineering**

Spring 2024

**Assignment 5: Mining issue trackers and version control data**

TEAM 13

Team Members: -

GAURAV SHARMA (GXS230001)

SHALIN RONAKKUMAR KAJI (SXK220263)

ADITYA KRISHNA (

**Analyzing issue tracker activity**

Q) How many issues have been reported in the project

5794

Q) How many of these issues are bugs, improvements, new features, tasks, wishes, sub-tasks, and tests? What are their percentages? How many issues of each of these types have been resolved or closed? What are their percentages? Fill in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type of issue | Total | | Resolved/Closed | |
|  | # | % (Round off) | # | % (Round off) |
| Bug | 4123 | 71.16% | 3886 | 67.07% |
| Improvement | 1061 | 18.31% | 958 | 16.53% |
| Task | 148 | 2.55% | 136 | 2.35% |
| New Feature | 236 | 4.07% | 215 | 3.71% |
| Sub-Task | 95 | 1.64% | 88 | 1.52% |
| Wish | 109 | 1.88% | 89 | 1.54% |
| Test | 22 | 0.38% | 22 | 0.38% |
| Total | 5794 | 100% | 5394 | 93.1% |

**Finding co-changed code files**

|  |  |  |
| --- | --- | --- |
| No of co-changed files | List of co-changed files | List of commits |
| 2 | pdfbox/src/main/java/org/apache/pdfbox/pdmodel/graphics/shading/Type1ShadingPaint.java  pdfbox/src/main/java/org/apache/pdfbox/pdmodel/graphics/shading/Type4ShadingContext.java  pdfbox/src/test/java/org/apache/pdfbox/pdmodel/TestFDF.java  pdfbox/src/test/java/org/apache/pdfbox/pdmodel/TestPDDocument.java  src/main/java/org/apache/pdfbox/util/operator/pagedrawer/ClipEvenOddRule.java  src/main/java/org/apache/pdfbox/util/operator/pagedrawer/ClipNonZeroRule.java  pdfbox/src/main/java/org/apache/pdfbox/pdmodel/graphics/shading/Type4ShadingPaint.java  pdfbox/src/main/java/org/apache/pdfbox/pdmodel/graphics/shading/Type5ShadingContext.java | 1. 493ca599868ef0a1b8bbe8d19d102282bef95ebf 2. 5e4fcd934a91f7109515f74e45b0ffeac5a5c372 3. b242c0bde7a7544407eb9119f7465ddcf476ea1e 4. d7be592bc0022cfa6fd3ffcdd5989b78bc827cb3 5. 18ef6491e301ef19559cd4c32a511b39a3f398ee 6. 28c08a6152230022a208caadc887bdc86ac1885e 7. 5546a3c7f8cd7d9995bfba7ee27a114973564042 8. 6a07dac3fb764a9f047667dc47453eced8259d1e 9. 7fd9987c4e265092602265acee633abecad322a3 10. 8e26cb6c67b066789da6c08519d8ac27d5c35992 11. 9234659b437de009d5222f512e9d8a03c7cd11d5 12. b4a050c725048d8067ccf90c873dffbf00e4ced2 13. 493ca599868ef0a1b8bbe8d19d102282bef95ebf1b26f 14. 4c3253c8ac6d55f0834cf41be3566248e57867d4 15. 5e4fcd934a91f7109515f74e45b0ffeac5a5c3726d9a2d 16. 87b737537fa4db17140bf6ad29e695b1fc451584 |
| 3 | pdfbox/src/main/java/org/apache/pdfbox/examples/signature/ShowSignature.java  pdfbox/src/main/java/org/apache/pdfbox/examples/signature/package.html  pdfbox/src/main/java/org/apache/pdfbox/examples/util/ExtractTextByArea.java  pdfbox/src/main/java/org/apache/pdfbox/filter/FlateFilter.java  pdfbox/src/main/java/org/apache/pdfbox/filter/IdentityFilter.java  pdfbox/src/main/java/org/apache/pdfbox/filter/JBIG2Filter.java  pdfbox/src/main/java/org/apache/pdfbox/contentstream/operator/graphics/FillEvenOddRule.java  pdfbox/src/main/java/org/apache/pdfbox/contentstream/operator/graphics/FillNonZeroAndStrokePath.java  pdfbox/src/main/java/org/apache/pdfbox/contentstream/operator/graphics/FillNonZeroRule.java | 1. 200b100383815e85366e2b06caa6ab62c2430633 2. 836cf16d600a985eecf1d190894f41d7034de8a9 3. b688014755d2f8d866486d022a3779c2e7b637f7 4. 1ed602dde338d78517e23cb8371775632323418a 5. daeee5e2ebaaee092ca53f9ab7f406136b2ff071 6. ed3f7e9070d9f4726baae7e49bc2ed7858b09de3 7. 34f86618e9abde7c29db6d05f7a56f18d76dc49f 8. b9273c34bf11968c3bf25122efd79197f04c8964 9. a8cbfd833f52da467b561b068cfca9f09751e48f |
| 4 | src/org/pdfbox/util/operator/pagedrawer/FillNonZeroRule.java  src/org/pdfbox/util/operator/pagedrawer/Invoke.java  src/org/pdfbox/util/operator/pagedrawer/LineTo.java  src/org/pdfbox/util/operator/pagedrawer/MoveTo.java  pdfbox/src/main/java/org/apache/pdfbox/contentstream/operator/graphics/ClosePath.java  pdfbox/src/main/java/org/apache/pdfbox/contentstream/operator/graphics/CurveTo.java  pdfbox/src/main/java/org/apache/pdfbox/contentstream/operator/graphics/CurveToReplicateFinalPoint.java  pdfbox/src/main/java/org/apache/pdfbox/contentstream/operator/graphics/CurveToReplicateInitialPoint.java | 1. 0043363995339865503811c70dd991b57c06b3cf 2. 623a0af37723377676f57213cf74b1a909c23272 3. d18984f70b1197cbf53e2e3f6472018b26ced7b5 4. d8915013d5466c6b6a5e2584bcc64e8ce469be7d 5. a8cbfd833f52da467b561b068cfca9f09751e48f 6. b9273c34bf11968c3bf25122efd79197f04c8964 7. fe69622beeb63f85388fe11074ce93c6765f544c |
| 5 | src/org/pdfbox/util/operator/SetStrokingCMYKColor.java  src/org/pdfbox/util/operator/SetStrokingColorSpace.java  src/org/pdfbox/util/operator/SetStrokingGrayColor.java  src/org/pdfbox/util/operator/SetStrokingRGBColor.java  src/org/pdfbox/util/operator/SetTextFont.java | 1. 0043363995339865503811c70dd991b57c06b3cf 2. 623a0af37723377676f57213cf74b1a909c23272 3. d18984f70b1197cbf53e2e3f6472018b26ced7b5 |

Select the set of 3 co-changed files that are most frequently changing together and select the set of 2 co-changed files that are most frequently changed together. Analyze the code and provide an explanation why these sets files change together so frequently. Feel free to measure the coupling between the classes in these files or even run a bad smell detector.

**Set of 3 co-changed files :-**

|  |  |
| --- | --- |
| pdfbox/src/main/java/org/apache/pdfbox/pdmodel/interactive/digitalsignature/PDSeedValueMDP.java  pdfbox/src/main/java/org/apache/pdfbox/pdmodel/interactive/digitalsignature/PDSeedValueTimeStamp.java  pdfbox/src/main/java/org/apache/pdfbox/pdmodel/interactive/digitalsignature/PDSignature.java | 1.2f3188babd3bdb5a2c25158b952164a97f9cbe50  2.565b8cb741520c529c340a9298870bb7ef0df8e9  3.bdad8a2d0b3ccf23e08b3488d91208ea4a73ccc7 |

Examining closely the given code samples, we can conclude that -the three classes (PDSeedValueMDP.java, PDSeedValueTimeStamp.java and PDSignature.java)-\ illuminate aspects of digital signatures relevant to PDF documents. Here are some reasons why these files might change together frequently:Here are some reasons why these files might change together frequently:

Common Functionality:

These two files are an integral part of PDF digital signatures so they should expect common functionalities or work with similar components of a PDF document.

Likewise, the PDSignature and PDSigDataType represent two different signature types or options. On the other hand, PDSeqNo, PDSignMsgURL, and PDSigIdiropct are closely related because they deal with message sequence numbers, mathematical operations, and various aspects of data integrity.

Interdependency:

The classes might have interconnections, therefore, the changes in the first class shall invoke changes in the other classes accommodating for continuity, identity and functionality.

Likewise, the version of field format or options (PDSignature) may require matching modification of the seed value optionwhether by time stamping (PDSeedValueMDP) or time stamping (PDSeedValueTimeStamp) to suit to the new characteristics or standards.

Evolution of PDF Standards:

The technological specifications governing PDF formatting and digital signatures are often being adapted in order to cope with emerging challenges and opportunities. Alerts and addition to PDFs template may b equally significant updates across multiple classes document that are digitally signed.

Likewise, if there was a new PDF standard that came out with different styles of signatures or specific data structure, the corresponded classes should be altered properly so that the new standard will be implemented fully.

Maintenance and Refactoring:

Taking up the exercise of refactoring of maintenance activities can cause the series of changes in related classes for imporving the structure, clarity or performance.

In addition to this, code refactorings of the way the frequently used functions will be put into shared components or interfaces too can cause ripple effects in other files.

In software systems, coupling describes the level of interdependence between modules or classes. Low coupling denotes slack links and little dependencies between components, whereas high coupling implies that classes or modules are highly dependent on one another. Now let's examine the coupling between the three classes (PDSignature, PDSeedValueTimeStamp, and PDSeedValueMDP) that have been supplied.  
Elevated Coupling:  
A system with a high coupling is one in which modifications in one class can significantly affect other classes, resulting in a highly coupled system. This may make it harder to test, maintain, and alter the system.  
Direct method calls across classes, sharing a large number of characteristics or dependencies, or strongly depending on one another's internal implementations are a few examples of high coupling.

**Set of 2 co-changed files :-**

|  |  |
| --- | --- |
| pdfbox/src/main/java/org/apache/pdfbox/pdmodel/fdf/FDFCatalog.java  pdfbox/src/main/java/org/apache/pdfbox/pdmodel/fdf/FDFDictionary.java | 1dd4d3eefa6691c79f1c4dda69198e9bbeccddc5  200b100383815e85366e2b06caa6ab62c2430633  3a66990e0edf7b60d45fc0ef55ea02a353127990 |

Two Java files, FDFCatalog.java and FDFDictionary.java, make up the given code. These files are closely linked, indicating a high level of dependency and frequent changes made jointly for a variety of reasons:  
  
Dependency Directed:  
  
FDFDictionary is directly dependent upon FDFCatalog. FDFDictionary objects are necessary for the operation of getFDF() and setFDF() in FDFCatalog.  
Moreover, FDFCatalog constructors and writeXML (Writer output) methods employ FDFDictionary.  
Connectivity of Function:  
  
The handling of Forms Data Format (FDF) documents is strongly related to both classes. An FDF dictionary is represented by FDFDictionary, and an FDF catalog by FDFCatalog. They collaborate to manage data and processes unique to the FDF.  
Comparable Reasoning:  
FDF data must be serialized to XML using methods like writeXML(Writer output) in FDFDictionary. FDFCatalog uses this XML document to write the FDF catalog.  
When creating an FDF object from an XFDF XML document, the FDFCatalog constructor also makes direct use of FDFDictionary.  
Information Sharing:  
  
FDFCatalog interacts with FDF dictionary data contained in FDFDictionary objects using functions like getFDF() and setFDF(FDFDictionary fdf).  
Insufficient Cohesion  
  
Due to their shared emphasis on FDF-related functions, both classes may be more closely coupled because they frequently collaborate to complete FDF document manipulation tasks.

The final outputs are added to the github :-

<https://github.com/gauravsharma2/SoftwarePdfBox/tree/main>

**Linking changes and issues**

Link the data from issue trackers and versioning control systems to answer the following questions:

Final output code and CSV for these questions too added on github :-

<https://github.com/gauravsharma2/SoftwarePdfBox/tree/main/PdfBox_Project/Linking%20changes%20and%20issues>

1. What are the commits addressing each issue in the issue tracker? Note that multiple commits can be made to address one issue :-

* The script **“test.py”** is prepared to link changes and issues
* The csv file **“solution.csv”** is prepared, containing the issue-id, commits, source files added, source files modified, source files deleted.
* It is also observed that there are certain issues in the issue-tracker whose commits are not recorded.

2. How many source files are added to address each issue :- 3592

3. How many source files are modified to address each issue :- 20024

4. How many source files are deleted to address each issue :- 1714

5. How many source files on average are added/modified/deleted to address issues :-

Added Modified Delete

|  |  |  |
| --- | --- | --- |
| 1.150544523 | 6.413837284 | 0.549007047 |

Contribution: -