Metrics Set 1 – Complexity Metrics

The complexity metrics measure the complexity of executable code within procedures. High complexity is more difficult to understand, and future code changes will take longer, so code development will be more complicated. Usually, high complexity is related to the high number of lines of code, but this is not always the case. There may be function with a few lines of code but with a lot of complexity.

The complexity metrics are sub divided in 5 different metrics each with different targets.

The first one, method metrics is composed by 4 metrics:

* CogC - Cognitive complexity.
* ev(G) - Essential cyclomatic complexity.
* iv(G) – Design complexity.
* v(G) – Cyclomatic complexity.

The second, class metrics is composed by 3 metrics:

* OCavg – Average operation complexity.
* OCmax – Maximum operation complexity.
* WMC - Weighted method complexity.

The last three, package metric, module metrics and project metrics are composed by 2 metrics:

* v(G)avg – Average cyclomatic complexity.
* v(G)tot – Total cyclomatic complexity.

For this analysis the chosen metrics are CogC, v(G) and iv(G), all from method metrics.

**CogC metric - cognitive complexity metric**

The cognitive complexity is used to know how difficult the code is to intuitively understand. Unlike Cyclomatic Complexity, which is used to know how difficult your code will be to test. For example, the cognitive complexity increases when there are breaks in the flow of the code.

Parts of code with higher cognitive complexity will be harder to read and understand.

Table 1 - Methods with the highest cognitive complexity:

|  |  |  |
| --- | --- | --- |
| Method | Value CogC | Source path |
| refreshCiteMarkersInternal  (List<BibDatabase> databases, OOBibStyle style) | 189 | src\main\java\org\jabref\gui\openoffice\OOBibBase.java |
| importDatabase  (BufferedReader reader) | 188 | src\main\java\org\jabref\logic\importer\fileformat\RisImporter.java |
| formatName  (Author author, String format, Warn warn) | 154 | src\main\java\org\jabref\logic\bst\BibtexNameFormatter.java |
| parseEntry  (Element e) | 135 | src\main\java\org\jabref\logic\importer\fileformat\GvkParser.java |
| importDatabase  (BufferedReader reader) | 131 | src\main\java\org\jabref\logic\importer\fileformat\BiblioscapeImporter.java |
| getEntryFromPDFContent  (String firstpageContents, String lineSeparator) | 128 | src\main\java\org\jabref\logic\importer\fileformat\PdfContentImporter.java |

In this table, cognitive complexity is ordered from largest to smallest so it can be observed that method *refreshCiteMarkersInternal(…)* is the one with the highest cognitive complexity, it means the method is difficult to read and understand and to modify in the future, probably with a high cognitive complexity there might be a code smell, for example a long method, or a large class because usually complexity is related to number of lines of code (not always).

**v(G) metric - cyclomatic complexity metric**

The cyclomatic complexity also known as McCabe's complexity, it is a count of the linearly independent paths through the code. In other words, simply the number of decisions that the code needs to make. Decisions are caused by conditional statements(if, for, while).

Table 2 - Methods with the highest cyclomatic complexity:

|  |  |  |
| --- | --- | --- |
| Method | Value v(G) | Source path |
| transformSpecialCharacter  (long c) | 148 | src\main\java\org\jabref\logic\layout\format\RTFChars.java |
| importDatabase  (BufferedReader reader) | 110 | src\main\java\org\jabref\logic\importer\fileformat\RisImporter.java |
| getDescription  (Field field) | 96 | src\main\java\org\jabref\gui\fieldeditors\FieldNameLabel.java |
| getSourceField  (Field targetField, EntryType targetEntry, EntryType sourceEntry) | 80 | src\main\java\org\jabref\model\entry\BibEntry.java |
| parseEntry  (Element e) | 79 | src\main\java\org\jabref\logic\importer\fileformat\GvkParser.java |
| importDatabase  (BufferedReader reader) | 76 | src\main\java\org\jabref\logic\importer\fileformat\BiblioscapeImporter.java |

In this table, cyclomatic complexity is ordered from largest to smallest so it can be observed that method *transformSpecialCharacter(…)* is the one with the highest cyclomatic complexity, it means that the method has a lot of conditions, cycles or functions calls so the method has many linearly independent paths, in this case 148, so there might be a code smell, for example a long method because how bigger the cyclomatic complexity is the more functions calls and conditions (if, for, while) it has.

**iv(G) metric – design complexity metric**

The design complexity is referent to how interlinked a methods control flow is with calls to other methods. Design complexity values vary between 1 to v(G) (cyclomatic complexity). Design complexity also represents the minimal number of tests necessary to exercise the integration of the method with the methods it calls.

Table 3 - Methods with the highest design complexity:

|  |  |  |
| --- | --- | --- |
| Method | Value iv(G) | Source path |
| importDatabase  (BufferedReader reader) | 102 | src\main\java\org\jabref\logic\importer\fileformat\RisImporter.java |
| getDescription  (Field field) | 96 | src\main\java\org\jabref\gui\fieldeditors\FieldNameLabel.java |
| parseEntry  (Element e) | 69 | src\main\java\org\jabref\logic\importer\fileformat\GvkParser.java |
| importDatabase  (BufferedReader reader) | 67 | src\main\java\org\jabref\logic\importer\fileformat\BiblioscapeImporter.java |
| getFieldValue  (BibEntry entry, String pattern, Character keywordDelimiter, BibDatabase database) | 55 | src\main\java\org\jabref\logic\citationkeypattern\BracketedPattern.java |
| refreshCiteMarkersInternal(List<BibDatabase> databases, OOBibStyle style) | 53 | src\main\java\org\jabref\gui\openoffice\OOBibBase.java |

In this table, design complexity is ordered from largest to smallest so it can be observed that method *importDatabase(..)* is the one with the highest design complexity. As said above, the design complexity is part of v(G) cyclomatic complexity, but focused more on the calls to other methods, so the highest design complexity, the method has more interconnections to other methods, and a higher number of methods calls.

**Summary**

All these metrics discussed above contribute to finding code smells about complexity, as said above the high complexity is related to the high number of lines of code normally, so it is related to the code smell long method. In this project some of my co-workers found this code smell, for example Martim Gouveia 57482.