SARPREET SINGH BUTTAR
5/10/18

ASSIGNMENT 2: QUALITY MODEL / INDIRECT METRICS

Introduction

The purpose of this report is to determine the *Maintainability* and *Re-Usability* of a Java library called *JSON Web Token for Java and Android (JWT)*[1]. This is the same project that we use in the previous assignment. Therefore, we use the same analysis results obtained during the previous assignment.

Top Level Package

JWT contains three sub projects named as *API, Extension*, and *Impl*. There are some packages in these projects that contains very few classes, i.e., < 16. Therefore, we merge the classes of these packages in a single package and named it as *others*. Table 1 summarizes top level package.

Package Name	Number of Classes
io.jsonwebtoken	29
others	28
io.jsonwebtoken.io	25
io.jsonwebtoken.impl.crypto	20
io.jsonwebtoken.impl	17

Table 1: Top Level Package

Metrics for Top Level Package

We follow the given software quality matrix to define the metrics for top level package. In particular, we select the metrics of *Analyzability* property. Table 2 shows the metrics for top level package.

Metric	Impact
LOC	Indirect
WMC	Indirect
DIT	Indirect
NOC	Indirect
DAC	Indirect
TCC	Direct
LOD	Indirect

Table 2: Metrics for Top Level Package

Methods to Find Outliers in a Package

We use *Python* programming language to implement methods that find the outliers in a package.

1. **Top 15 classes:** This method sorts the classes of each package based on the impact of a given metric. For instance, if the impact is indirect, the classes are sorted in descending order, whereas for direct it is vice-versa. Once the sorting is done, it marks

[1] https://github.com/jwtk/jjwt

the first 15 classes with value 1, and remaining with value 0. The 1 and 0 represent the *inverse metric* value of a class. In excel sheet, it is denoted as single quotation mark, such as NOC’. Later, it adds the inverse metrics of each package for a given metric to get the aggregated metric, which is denoted with double quotation mark, such as NOC’’. In the end, it divides the aggregated metric with the number of classes in a given package to get the *high-level metric*, which is denoted with triple quotation mark, such as NOC’’’.

2. **Top 15% classes:** This method has same procedure as the above method. The only difference is rather than selecting top 15 classes, it selects top 15% classes.

3. **Top 15% classes in a range:** This method has also same procedure as the above methods. Again, the difference is in selecting the classes. In this case, if the impact is direct, it selects those classes whose given metric value is in the following range:

   $$min <= given\_metric\_value <= max + 15\% \ (max - min)$$

   Here, min and max represent minimum and maximum metric values in a specific package. Similarly, if impact is inverse, it follows the following range:

   $$max – 15\% \ (max - min) <= given\_metric\_value <= max$$

Maintainability and Re-usability
We also use *Python* programming language to compute the *main* and *sub properties*, based on given software quality matrix, of JWT. According to given software quality matrix, the sub properties *weights* are denoted with ‘++’, ‘--', ‘+’, and ‘-’. Here, ‘++’ or ‘--' is 2, whereas ‘+’ and ‘-’, is 1.

We compute the sub properties by multiplying the high-level metric, i.e., X’’’, with the weight of a given sub property. In addition, we divide the results with the weights to *normalize* the value. For instance:

$$Analyzability = \frac{2*LOC''' + 2*WMC''' + 2*DIT''' + 1*NOC''' + 2*DAC''' + 2*TCC''' + 2*LOD'''}{2 + 2 + 2 + 1 + 2 + 2 + 2}$$

Once the sub properties are computed, we calculate the main properties by adding their sub properties. For instance:

Re-Usability = Understandability + Learnability + Operability + Attractiveness

Maintainability = Analyzability + Changeability + Stability + Testability

Note, these values are computed for each of the methods that we mentioned earlier.

Ranking Based on Re-Usability

Package Name	Re-Usability
io.jsonwebtoken.impl	3.52941176470588
io.jsonwebtoken.impl.crypto	3
io.jsonwebtoken.io	2.4
others	2.14285714285714
io.jsonwebtoken	2.06896551724138

Table 3: With Top 15 Classes Method

Package Name	Re-Usability
io.jsonwebtoken.impl	0.705882352941176
io.jsonwebtoken.io	0.64
io.jsonwebtoken.impl.crypto	0.6
others	0.571428571428571
io.jsonwebtoken	0.551724137931034

Table 4: With Top 15% Classes Method

Package Name	Re-Usability
io.jsonwebtoken.impl.crypto	1.39120879120879
io.jsonwebtoken.impl	0.862843039313628
io.jsonwebtoken.io	0.801198801198801
io.jsonwebtoken	0.649109511178477
others	0.580348223205366

Table 5: With Top 15% Classes in a Range Method

Ranking Based on Maintainability

Package Name	Maintainability
io.jsonwebtoken.impl	3.52941176470588
io.jsonwebtoken.impl.crypto	3
io.jsonwebtoken.io	2.4
others	2.14285714285714
io.jsonwebtoken	2.06896551724138

Table 6: With Top 15 Classes Method

Package Name	Re-Usability
io.jsonwebtoken.impl	0.705882352941177
io.jsonwebtoken.io	0.64
io.jsonwebtoken.impl.crypto	0.6
others	0.571428571428571
io.jsonwebtoken	0.551724137931034

Table 7: With Top 15% Classes Method

Package Name	Maintainability
io.jsonwebtoken.impl	1.54249084249084
io.jsonwebtoken.io	0.967966673849027
io.jsonwebtoken.impl.crypto	0.944664224664225
others	0.81857605995537
io.jsonwebtoken	0.735609628466771

Table 8: With Top 15% Classes in a Range Method

Conclusion

The tables shows that according to all the methods, *io.jsonwebtoken.impl* package has the highest maintainability, whereas *io.jsonwebtoken* package has the lowest. It is worth to mention that *io.jsonwebtoken.impl* package has the maximum classes, whereas *io.jsonwebtoken* package has the minimum. Therefore, it is not good sign for JWT, since maximum effort is required to maintain the project. On the other hand, according to 15 classes and 15% classes methods, *io.jsonwebtoken.impl* package has the highest re-usability, whereas *io.jsonwebtoken* package has the minimum. However, according to 15% classes in a range method, *io.jsonwebtoken.impl.crypto* package has the highest re-usability, and *others* package has the lowest. Again, this is not good for JWT, since most of classes are not re-usable.