|  |
| --- |
| **Concrete Architecture Evaluation Report (T1)**  Delivery Date: Apr 25, 2023  **Team Members:**  *Nurten ALBAYRAK ERGEN, Merve GÖNCÜ, Tahir ARPACIK*  *<Please enter your Team Members with student numbers and names.>* |

1. **OSS PRODUCT OVERVIEW**

**Apache/dubbo**

*<Describe the OSS product that you selected for your project work.>*

**Table 1.** Description of the OSS product

|  |  |
| --- | --- |
| Name of OSS Product: | **Apache/dubbo** |
| Description of OSS product: |  |
| URL of OSS product: |  |
| Size of OSS product (KLOC): |  |

1. **TOOLS USED FOR ANALYSIS**

*<List the tools that you used in the analysis of the OSS product.>*

**Table 2.** The list of tools used for analysis

|  |  |  |
| --- | --- | --- |
| **Tool Name** | **Purpose of Use** | **URL** |
| Understand by SciTools | *To obtain Maintainability metrics(CC metrics,NNL,*  *To create dependency graph* | https://scitools.com/feature/metrics |
| MetricReloaded | *to obtain maintainability metrics(C&K metrics,LOC* | https://github.com/BasLeijdekkers/MetricsReloaded |
| Sonarqube |  | https://www.sonarsource.com/products/sonarqube/ |

1. **GQM TREE FOR EVALUATING OSS MAINTAINABILITY**

*<Summarize the goal and the questions to evaluate the maintainability of the OSS product by software metrics. It is suggested that your questions are answered by some indicators such as graphs (figures) or tables.>*

**Table 3.** The goal, questions, and metrics to evaluate OSS maintainability

|  |  |  |
| --- | --- | --- |
| **Goal:** | To understand, the maintainability of the OSS product, from the viewpoint of *<architect/developer>* | |
| *List of questions to answer while evaluating the goal:* | | *List of metrics used to answer the question:* |
| **Q(1):** |  |  |
| **…** |  |  |
| **Q(n):** |  |  |
| **Goal:** | facilitate the system's maintainability, from the viewpoint of developer/tester | |
| **Q1:** | What is the current maintainability index of the system? | Maintainability index  M1.1 : HV (Halsteadvolume)  M1.2: CC ()  M1.3: LOC (Line of Code) |
| **Q2:** | How can the testability of classes be evaluated? | M2.1: RFC  M2.2: NOC  M2.3: NNL |
|  | How can the system be evaluated in terms of interchangeability? (sistem değiştirilebilirlik açısından nasıl değerlendirilebilir) | M3.1: DIT  M3.2: CBO  M3.3: LCOM |
|  | How can the system be evaluated in terms of analysability? (sistem analiz edilebilirilik açısından nasıl değerlendirilebilir) | M4.1 : WMC  M4.2: CC |

1. **DESCRIPTION OF SOFTWARE METRICS**

*<Provide a unique list of software metrics to answer the questions in your GQM tree, each with its name, type, source (the name of the tool that provides value for it), and formula/description provided by the source.>*

**Table 4.** List of software metrics used in analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric Name** | **Type** | **Source** | **Formula (or description) by Source** |
|  | *<e.g., length, complexity, C&K, code smell>* |  |  |
| HV | Halstead |  |  |
| CC | *Complexity* | *Understand* | *Cyclomatic complexity.McCabe Cyclomatic complexity as per the original NIST paper on the subject. The cyclomatic complexity of any structured program with only one entrance point and one exit point is equal to the number of decision points contained in that program plus one. Understand counts the keywords for decision points (FOR, WHILE, etc) and then adds 1. For a switch statement, each 'case' is counted as 1. For languages with Macros, the expanded Macro text is also included in the calculation. (Formula:case,catch,do,for,if,?,while+1)* |
| LOC | lenght |  |  |
| RFC | *C&K* | *MetricReloaded* | *Calculates the Response For Class value of the class. This is defined as the number of possible methods that may be called when a class is sent a method send. In practice, this is the sum of the number of methods and constructors in the class, plus the number of methods and constructors that the class may directly call. Methods from or called by superclasses are not counted. Classes with high Response For Class have a higher complexity, may be less stable, and require higher amounts of integration testing.* |
| NOC | *C&K* | *MetricReloaded* | *Calculates the total number of direct subclasses of each class that occur in the project.* |
| NNL | *Complexity* | *Understand* | *Maximum nesting level of control constructs (if, while, for, switch, etc.) in the function.* |
| DIT | *C&K* | *MetricReloaded* | *Calculates the depth of the inheritance tree for each class. The depth is calculated as the number of inheritance steps between the class and java.lang.Object.* |
| CBO | *C&K* | *MetricReloaded* | *Calculates the number of classes or interfaces which each class is "coupled" with. A class is declared to be coupled with another if it depends on that class or is depended on by that class. Dependencies due to inheritance are not counted.* |
| LCOM | *C&K* | *MetricReloaded* | *Calculates on the degree of cohesiveness of a class. We use a variant of the LCOM metric designed by Hitz and Montazeri, which is more appropriate for Java. The metric says that two methods of a class are related if they share a variable use, or one method calls another. The metric is then the count of the number of components of the method relation graph. A value of 1 indicates a highly cohesive class, which can not easily be split into smaller classes. Higher values may indicate that the class may be "doing too much", and should be split. Note that constructors, equals(), hashCode(), toString(), clone(), finalize(), readObject(), and writeObject() methods are not considered, as these scaffolding methods often touch all variables in a class, and would thus result in metrics values indicating more cohesiveness than is actually apparent in the design.* |
| WMC | *C&K* | *MetricReloaded* | *Calculates the total cyclomatic complexity of the methods in each class.* |
|  |  |  |  |
|  |  |  |  |

1. **EVALUATION OF MAINTAINABILITY**

*<Please evaluate the maintainability of the OSS product in relation to the questions in your GQM tree.*

* 1. **Answer to <Question-1>**

***Kalıtım ağacının derinliğine bakıldığında (DIT) değerinin 1,87 olduğu görülmektedir. DIT metriğinin sayısı ne kadar yüksek olursa kalıtım ağacının derinliği de yüksek olur. Derinlerde bulunan sınıflarda ki metot sayısı yüksek olacağından kod değiştirilebilirliği zorlaşır ancak incelediğimiz sistemin DIT metriği yüksek değildir bu sebeple değiştirilebilirliğe açıktır.***

*Looking at the depth of the inheritance tree, it is* ***seen*** *that the (DIT) value is 1.87. The higher the number of DIT metrics, the higher the depth of the inheritance tree. Since the number of methods in the deep classes will be high, code changeability becomes difficult, but the DIT metric of the system which we examined is not high, so it is open to changeability.*

***CBO metriği aralarında kalıtım ilişkisi olmayan ancak birbirlerinin özelliklerini ve metotlarını kullanan sınıfların sayısını ölçer. Sınıflar arasındaki bağımlılığın yüksek olması değişebilirliği zorlaştırır. İncelediğimiz sistemin CBO değeri 9,81dir. Bu metrik değer belirlenen eşik değerinden yüksektir bu sebeple alt modülleri incelememiz gerekir.Alt modüllerin CBO değerlerine bakıldığında sadece dubbo-cluster ve dubbo-common adlı iki modülün değerin üstünde olduğunu,ortalamayı yükselttikleri görülmektedir.(dubbo-cluster(9,57),dubbo-common(13,09)) Sistem bu durumuyla değiştirebilirliğe açık değildir.***

*The CBO metric measures the number of classes that do not have an inheritance relationship but that use each other's properties and methods. The high dependency between classes makes changeability**difficult. The CBO value of the system which we examined is 9.81. This metric value is higher than the determined threshold value, so we need to examine the sub-modules. When the CBO values of the sub-modules are examined, it is seen that only two modules called dubbo-cluster and dubbo-common are above the value and increase the average**(dubbo-cluster(9,57),dubbo-common(13,09))The system is not modifiable with this state.*

***LCOM metriği metotlar arasındaki uyumsuzluğu ölçer. Metotlar arasında uyum yüksek olursa LCOM değeri daha düşük, uyumsuzluk yüksek olursa LCOM değeri daha yüksek olur. Değiştirebilirlik açısından incelediğimizde uyumun yüksek olması karmaşıklığın düşük olmasını sağlayağından değişebilirliği olumlu yönde etkileyecektir. İncelediğimiz sistemin LCOM değeri 2,44dür. Eşik değerler aralığında bulunduğundan değiştirilebilirliğe açıktır.***

*The LCOM metric measures cohesiveness between methods. If the cohesiveness between the methods is high, the LCOM value will be lower, and if the cohesiveness is high, the LCOM value will be higher. When we examine it in terms of changeability, high cohesion will have a positive effect on mutability as it will ensure low complexity. The LCOM value of the system we examined is 2.44. Since it is within the threshold values range, it is changeable.*

**5.n. Answer to <Question-n>**

*<If your evaluation is not at the product level but at lower levels of design elements (e.g. at package level), then provide your results accordingly by adding subsections (e.g. per package) as necessary.>*

**6.DEPENDENCY GRAPH(S)**

*<Please provide the dependency graphs(s) with respect to the concrete architecture of the OSS product.>*

*<If your graph(s) is not at the product level but at lower levels of design elements (e.g. at package level), then provide your results accordingly by adding subsections (e.g. per package) as necessary.>*

**7.OVERALL EVALUATION OF CONCRETE ARCHITECTURE**

*<By considering the information that you provide in sections 5 & 6, evaluate the concrete architecture of the OSS product in the overall. It is suggested that you consider the similarities in the results of the sections 5 & 6 in this evaluation, and also discuss any differences in the results.>*

**8.REFERENCES**

[1] <*MUST-DO: Please create a Google Sheet, where you provide values to the metrics that you used in this report, as a data store of your project work. Then provide link to that sheet as read-only, in this first reference.*>

*<Provide any other references that you used as input to your project work and report.>*

**9.ALLOCATION OF RESPONSIBILITIES WITHIN TEAM MEMBERS**

*<For groups of 2-3 people, please describe the allocation of responsibilities within team members.>*

*<You may allocate responsibilities with respect to the questions in your GQM tree, to the sections of this report, or to the design elements (e.g. packages) in your product architecture. In any case, provide a detailed explanation to objectively give credit to each member’s work.>*

|  |  |  |
| --- | --- | --- |
| **Name of Team Member** | **Description of Responsibility** | **Allocation Unit** |
|  |  | *<e.g., question #, section #, package # or name>* |

**APPENDIX. Bonus Part for T1 (10 pts)**

*<Please answer the following questions by including both quantitative and qualitative evidence as necessary.>*

**A1.** What are types, numbers, and locations of the code smells detected in the source code of the OSS product?

**A2.** Do you observe any relation(s) between the maintainability of the source code analyzed & the types, numbers, and locations of the code smells identified? If yes, explain the relation(s)?