**CSE 3063 OOSD FALL 2020 - PROJECT EVALUATION PLAN**

In order to evaluate your projects at each iteration you need to collect and report detailed statistics about your code. These statistics are listed below. Please use "CSE3063 Fall2020 Project Evaluation Metrics.ods" file to report.

It is crucial for you to provide all the statistics correctly. In your demonstrations we can check these and each mismatch will be penalized by -10 points.

**DOCUMENTS**

1- Software Requirement Specification (SRS) document {boolean: 0/1}
   If you provide a SRS document in this iteration or not
2- Domain Model (DM) {boolean: 0/1}
   If you provide a DM document in this iteration or not
3- Design UML Class Diagram (DCD) {boolean: 0/1}
   If you provide a DCD document in this iteration or not
4- Design UML Sequence Diagram (DSD) {boolean: 0/1}
   If you provide a DSD document in this iteration or not

**DESIGN**

5- Number of associations {numeric}
   Report how many associations you have between your domain classes in your UML Class Diagram (DCD).
6- Number of one-to-one associations {numeric}
7- Number of one-to-many associations {numeric}
8- Number of many-to-many associations {numeric}
9- Number of aggregations {numeric}
   Report how many aggregation associations you have between your domain classes in your UML Class Diagram (DCD).
10- Number of compositions {numeric}
    Report how many composition associations you have between your domain classes in your UML Class Diagram (DCD).
11- Number of generalization (is-a) relationships {numeric}
    Report how many generalization or in other words inheritance relationships in your UML Class Diagram (DCD). If there are several subclasses inheriting from a parent class, count each one of these. In short count all the subclasses.
12- Number of interface implementing classes {numeric}
    Report the number of domain classes that are implementing one or many interfaces in your UML Class Diagram (DCD).

**IMPLEMENTATION - CLASSES**

13- Number of classes {numeric}
    Number of domain classes in your project in your GitHub repository. Domain classes are the classes created by you to solve the problem. Domain classes do NOT include user interface or database connection classes. Classes from libraries do NOT count. Please note that each class should be a separate java file so that you should easily show this in your demos. This is an important grading criteria.
14- Number of abstract classes {numeric}
    Number of abstract classes among your domain classes in your project.

15- Number of interfaces {numeric}
Number of interfaces among your domain classes in your project.

**IMPLEMENTATION - ATTRIBUTES**

16- Number of reference type attributes {numeric}
Number of reference type attributes to your domain classes. Please see [13] for the definition of domain classes. Primitive type attributes such as int, double, Integer, Double and String attributes do NOT count. Also public attributes do NOT count. Only private and protected attributes whose type is one of your domain classes count. This is an important grading criteria.

17- Average number of reference type attributes {numeric}
Total reference type attributes in your domain classes/number of domain classes. Please see [16] for definition of reference type attributes.

18- Minimum number of reference type attributes {numeric}
Among your domain classes find the class with minimum number of reference type attributes and report the number of reference type attributes in this class.

19- Maximum number of reference type attributes {numeric}
Among your domain classes find the class with maximum number of reference type attributes and report the number of reference type attributes in this class

20- Rate of reference type attributes {numeric}
(Reference type attributes) / (total number of attributes including primitives and Strings) in your domain classes. Please see [16] for definition of reference type attributes. This is an important grading criteria.

**IMPLEMENTATION – METHODS (**DO NOT COUNT GET/SET METHODS**)**

21- Number of methods {numeric}
Total number of methods in your domain classes including methods with no parameters or with primitive type parameters such as int, Integer, double, Double, String, etc. Only the methods in your domain classes count.

22- Number of methods with reference type input/output parameters {numeric}
In your domain classes, count ONLY the methods with reference type input/output parameters. This is an important grading criteria.

23- Average number of methods in your domain classes {numeric}
(Number of methods [21]) / (number of domain classes)

24- Minimum number of methods in your domain classes {numeric}
Find the domain class which has the smallest number of methods (As always do not count get/set methods). Report the number of methods in this class.

25- Maximum number of methods in your domain classes {numeric}
Find the domain class which has the largest number of methods (As always do not count get/set methods). Report the number of methods in this class.

26- Rate of methods with reference type parameters {numeric}
[22] / [21]

**IMPLEMENTATION - CODE**

27- Total Lines of Code (LOC) {numeric}
The total lines of functional code you wrote in your domain classes. Empty lines, comments, lines in main function, and library includes do not count.

28- Average class LOC {numeric}
[27]/[13]
29- Minimum class LOC {numeric}
LOC in your smallest domain class. Please see [27] for how to count lines of code.
30- Maximum class LOC {numeric}
LOC in your largest domain class. Please see [27] for how to count lines of code.

**CODE EXECUTION**

31- Proper Execution with standart input {boolean: 0/1}
If the code executes without error and produce proper output json file given one of the standart input json file provided in the classroom.
32- Proper Trace and Logging {boolean: 0/1}
If the code outputs its execution trace to command line in addition to a log file or not. This is an important grading criteria.
33- Proper Execution with extended input {boolean: 0/1}
If the code executes without error and produce proper output json file given one of the extended input json file that is not given to the students. This field will be filled by your instructor or TA during your demo.

**DIFFERENCE WITH PREVIOUS ITERATION (**DO NOT COUNT GET/SET METHODS**)**

34- Lines of code (LOC) added {numeric}
(previous iterations LOC [27]) – (this iterations LOC [27])
35- Number of changed classes {numeric}
36- Number of changed methods {numeric}
37- Number of newly added domain classes {numeric}
38- Number of newly added methods {numeric}
39- Number of newly added attributes {numeric}
40- Number of deleted classes {numeric}

**RDD, HIGH COHESION/ LOW COUPLING, DESIGN PATTERNS**

41- LOC in Main function {numeric}
Lines of code in your main function. Please see [27] for how to count lines of code.
42- Class Size LOC Entropy {numeric}
Calculate distribution of code into your domain classes using Entropy formula. For example if you have three domain classes with 18, 22, and 60 LOC respectively, the entropy will be -18/100*LOG(18/100;3)-22/100*LOG(22/100;3)-60/100*LOG(60/100;3)=0,86. As log base use the number of domain classes so that it will give you a value between 0 and 1.
43- Method Size Entropy {numeric}
Similar to the [42], instead of using LOC for your unit, use the number of methods in each class. Please note that according to high cohesion design pattern we want the code to be distributed to your domain classes as much as possible in a meaningful way considering other GRASP patterns. So the entropy higher is the better.
44- Number of Polymorphic methods {numeric}
Count the polymorphic methods. For example in your text book, in Monopoly game, if Square class/interface landedOn abstract method is implemented in let's say four different classes, then this number will be 4.
45- Number of unique design patterns applied

Count the different design patterns (e.g. controller, singleton, creator, polymorphism, …) in your DCD and of course in your code. Please note that in your demo you need to show that these patterns applied in a meaningful way. For all these metrics, if you can't justify these numbers in your demo you will get –10 points for each wrong metric.