Project Phase 03

Software Testing - Graph Coverage

How to succeed with the project?

The project task sheets for each phase are published on the course homepage "Lecture: Software Testing" at StudIP. As described in the first lecture, each successfully completed project earns you points that sum up to the total score (applying the "5 out of 6" rule) which determines your final grade. **Cheating does not help you - but we will!**

How to complete a project phase successfully?

In order to complete a project phase successfully, you must upload your answers on StudIP. This can be done before **the main deadline** printed on the exercise sheet for a potential full amount of points, or before **the second deadline** for a reduced number of points. We will not consider any solutions handed in after the deadline! Furthermore, you must create the solutions to your project contributions **on your own**, and your test suite **must compile**. We encourage you to actively discuss all remaining problems in your groups, and - if that does not help - with one of the project advisors.

Technically, the project files you submit on StudIP must have the format as specified at the end of this task sheet. Furthermore, we will consider your last submission only. Therefore, if your first submission works as intended, and a second one does not, you will only get points based on the latter upload. Make sure your last submission is working!

How to get additional information?

We encourage you to discuss past and present project tasks, difficulties, and ideas with us. You can approach us during the project session, or during the weekly "office" hours.

MAIN DEADLINE:	23:59, May 18, 2022 UTC: 2022-05-18T23:59:00+02:00
SECOND DEADLINE:	23:59, June 01, 2022 UTC: 2022-06-01T23:59:00+02:00

The concepts of **Graph Coverage** are the main topic of the third project phase. Within these tasks, we will focus on the code-based version of graph coverage and data flow criteria, and - as usual - follow the triple division of basic questions, a sample case, and the application of graph-based coverage criteria to your team project.

To support you with the actual coverage determination, the tool *EclEmma* is used, which was once the Eclipse plugin implementation of the well-known *Emma* code coverage tool for Java, but is now fully based on the code coverage library *JaCoCo*. Due to its Eclipse plugin nature, it can be easily integrated into its normal workflow of test case creation and execution.

Setting Up EclEmma

a) In Eclipse, go to `Help` → `Eclipse Marketplace...`

b) Search for `EclEmma`

c) Install *EclEmma Java Code Coverage* (preferably version 3.1.3), if not already installed

d) Right-Click a *.java file containing JUnit tests, then `Coverage As` → `JUnit Test`

e) The results are found in the "Coverage" tab (Click on the three dots in the top right corner of the tab to switch between the Instruction, Branch, Line, Method, and Type counters, as well as cyclomatic complexity)

f) Take a look at the user guide[1] for more information on the features and handling of *EclEmma*

Graph Coverage - Tasks

Task 1 - Answer basic questions on Graph Coverage [3 P]

In this first task, you will do some basic research on the most important terms and criteria in the domain of graph coverage. For that purpose, answer the following questions in 2-3 sentences each:

a) Define the following terms in your own words:
   1) Graph
   2) (Test-)Path
   3) Syntactic and Semantic Reach

b) Which testing situations are suitable for the Graph Coverage approach?

c) What is the difference between *Tours*, *Tours With Sidetrips*, and *Tours With Detours*?

d) Describe (1-2 sentences) the *Node Coverage* (NC) and *Edge Coverage* (EC) criterion. What are their counterparts for code-based coverage?

[1] http://www.eclemma.org/userdoc/index.html

e) Name and describe (2-3 sentences) 2 *Path Coverage Criteria* OR 2 *Data Flow Test Criteria*. Does one of these two criteria subsume the other?

***Submission**: **.pdf file with your answers*

Task 2 - Apply Graph Coverage criteria to a sample program [5 P]

In the following, you will analyze a sample program with respect to its control flow, and create test cases that fulfill specific coverage criteria. The function of interest in this task is an excerpt of a constructor routine for the HashMap class, as found in a previous OpenJDK 6 implementation[2]. Solve the following tasks for the code section shown in Figure 0.1.

```
1  import java.util.Map.Entry;
2
3  public class HashMap<K,V> {
4          static final int MAXIMUM_CAPACITY = 1 << 10;
5          transient Entry<K,V>[] table;
6          int threshold;
7          final float loadFactor;
8
9      public HashMap(int initialCapacity, float loadFactor) {
10         if (initialCapacity < 0)
11             throw new IllegalArgumentException("Illegal initial capacity: " +
                   initialCapacity);
12         if (initialCapacity > MAXIMUM_CAPACITY)
13             initialCapacity = MAXIMUM_CAPACITY;
14         if (loadFactor <= 0 || Float.isNaN(loadFactor))
15             throw new IllegalArgumentException("Illegal load factor: " + loadFactor);
16
17         // Find a power of 2 >= initialCapacity
18         int capacity = 1;
19         while (capacity < initialCapacity)
20             capacity <<= 1;
21         this.loadFactor = loadFactor;
22         threshold = (int)(capacity * loadFactor);
23         table = new Entry[capacity];
24     }
25
26     public int getCapacity() {
27         return table.length;
28     }
29 }
```

Fig. 0.1: Sample code of the HashMap constructor method[2]

a) Create the control flow graph for the given constructor.

b) Create a minimum set of test cases that reaches 100% coverage for the instruction coverage criterion.

c) Extend the set of test cases so that it additionally reaches 100% coverage for the branch coverage criterion. Describe the necessity of the added tests.

[2] http://hg.openjdk.java.net/jdk6/jdk6/jdk/file/8deef18bb749/src/share/classes/java/util/HashMap.java

d) Analyze the code regarding the following data flow criteria, and list all relevant DU pairs. Does your test suite require additional tests to cover them?

   1) All-defs with respect to `capacity`
   2) All-uses with respect to `loadFactor`

***Submission**: *.pdf file with your control flow graph, *.pdf file with answers, *.java file with test implementations, *.jpg of a sample screenshot showing the EclEmma results of one subtask*

Task 3 - Apply Graph Coverage criteria to your software project [8 P]

You will now apply the concepts of graph coverage to your selected software projects. For the extension of your test suite, you are allowed to choose between two different tasks, depending on your current project focus and findings. During this task, EclEmma is again the recommended tool for the coverage analysis, but you are not bound to it, especially in case that you want to work with specific coverage criteria that are not supported by this tool.

a) Measure the coverage of your given project test suite (which includes the existing test suite as well as the tests that you created in previous project phases) by three graph coverage criteria which you can freely choose. Describe each individual result in 2-3 sentences.

b) Extend the test suite with own tests, which have to fullfil **one** of the following criteria:

   1) Increase the coverage values of all three coverage criteria that you applied in the previous subtask with at least 10 tests (compare and describe the effects on coverage for each individual test), OR
   2) Reveal a new bug in the software project (describe the bug, its context, and a potential fix in detail)

(NOTE: Make sure that you pick coverage criteria that do not already reach a coverage of 100%, so there is some room for your improvements)

***Submission**: *.pdf file with answers, *.java file with test cases*

NOTE: The task solutions for this phase should be uploaded as a `*.zip` file to the folder `project_uploads/phase_03/group_XX` on StudIP, using the name convention `Solution_Phase03_[StudentName]`, followed by an optional `_V[VersionNumber]` in case that you submit multiple versions of your solutions.

Example: `Solution_Phase03_SaschaLehmann_V1`

Please use the given LaTeX file `project_phase03_template.tex` for your solutions (also add your relevant java code parts to the indicated sections) and compile it into `*.pdf` format. Additionally, do not forget to include your actual java code files (`*.java`) in the `*.zip` file as well.