Interpret the metrics offered by the static analyzer

Assignment start date: 05th of March 2016, 00:00
Assignment due date: 14th of March 2016, 23:59
Format of the delivery: PDF
Expected length of the delivery: 4-10 pages
Pass criteria: You have to score minimum 1 point to be able to attend the exam. You can score maximum 25 points.

Problem description:
To improve performance, maintainability and readability of a program, static analysis proves to be an invaluable tool. It can tell us about which areas of the code to rewrite, eliminate, decouple, etc, so that the overall performance of the code improves.
This kind of testing is very common for software developers and architects, and can give you important practical knowledge.

Your task will be to discover what a source code is supposed to do, to manually test it and to analyze the source code with the help of a static analyzer. You will have to explain the metrics that you get. You will have to modify the source code of the project and measure it again, in order to improve the metrics. You will have to explain and compare the initial metrics with the final ones.

You will choose the source code from a total of 8 already developed projects: two C++, three CSharp, and three Java. You will chose just one project, the one who suits your programming language knowledge best or the one you would like most to analyze. You will find in the course webpage the zipped attachment containing all projects.

Work environment:
You will have to set-up your work environment as follows:

Source Code Analyzer: Download and install **Source Monitor**. It works only on Windows (7,8, 10).*
http://www.campwoodsw.com/sourcemonitor.html.

Development Environment: Use any IDE you prefer: **Visual Studio**, **Visual C++, Eclipse, IntelliJ** etc.

Source Control System: **Github**. You will have to create a free account and upload your modified projects there. https://github.com/

** If you have a Mac computer, you can use any Windows emulator to run Source Monitor (ex: Parallels software). Or you can use the computers at the university, which are running Windows. In order to install Source Monitor on them, you just have to unzip the Source Monitor files provided as attachment and copy them on the computer you will work on - you do not need special rights to do so.*

<u>Your assignment</u>

Step 1:
Choose from the provided package a project at your choice. It can be wither C++, CSharp or Java. You don't need to choose more than one project.

Step 2:

Source Monitor needs its to have its own project in order to work. So you need to create one first (.smproj) In Source Monitor: File -> New project:

- Select the programming language of the source code to be analyzed
- Give the Source Monitor project a name and a location
- Browse and select the source code that you want to analyze. Select the whole project.
- <u>Tick "Allow parsing of UTF-8 files</u>"
- Finish creating the new Source Monitor project

<u>Requirement 1 (5p):</u>

Make a brief description of the program that you use for your project assignment.

Make an analysis of the testable parts of the program that you are using for the project assignment. Design the manual tests that you would perform for this program, in order to test its functionality (system tests). Specify if you used a particular test design technique in designing your tests.

Would it make sense or not to write non-functional tests for the chosen source code? Motivate your choice.

Write a list with the concrete test cases that you obtained, as a result of the analysis and design performed above. The list needs to contain: pre- and post- test conditions (if the case), the test itself, the expected result and the actual result. Order the list of tests based on the importance of the tests. Can you tell something about your how did your previous knowledge / experience helped you in creating this list with tests?

<u>Requirement 2 (10p):</u>

Note that in SourceMonitor you can analyze metrics at both project level and at file level. We will need to do both for this project assignment.

Metrics at project level

Run the static analyzer on the whole project to be analyzed: In Source Monitor main window, select the checkpoint name, go to "View" menu and choose "Display checkpoint metrics summary". Select the checkpoint name, go to "View" menu and choose "Display **checkpoint** metrics Kiviat graph".

Take a screenshot of both the metrics summary an of the Kiviat graph. Write the following code analysis observations and interpretation:

- What metrics do you spot for the whole project in the window Metrics Summary for Checkpoint? Write a brief description of the metrics. Try to explain their values (below what is expected, as expected or above the expected level). What metrics do you think need to change?
- Which is the biggest file you have in your project by the number of lines of code?
- Which is the file with most branches in your project?
- Which is the file with most complex code? What metric did you choose to answer to this question?

Metrics at file level

Now, do the same with a significant file from your project (choose a file from the projects, select it, go to "View" menu and choose "Display **file** metrics Kiviat graph"). Take a screenshot of your Kiviat graph.

Write the following code analysis observations and interpretation:

- How do you interpret the metrics applied on your file? How are they different the metrics you obtained on the whole project, compared with the metrics on this file?
- Would you refactor (re-write) any of the methods you have in this file?

<u>Requirement 3 (10p):</u>
Improve the code, based on the metrics you have obtained with this analyzer. Analyze the improved code using SourceMonitor again.

- Identify at project level the metrics that you need to improve
- Perform the code changes accordingly. The code changes can be done for one or several files.
  - Provide in the delivery examples of improved/refactored code.
  - What are the results of running the analyzer on the improved code? Compare the newly obtained metrics with the old ones, both at file level and at project level. Comment on the newly obtained metrics.

    Note: on your SourceMonitor project, you can create many branches that will allow you to see how the code improves. Therefore, you are able to make improvements and measure them in consecutive steps.
- Upload the modified project in Github. In your project assignment, provide the web link to your modified code.
- In the end, make a couple of remarks about how easy or not it was for you to maintain the code (to modify it in order to improve it). Is there anything that you would have done differently on the initial code to make its maintenance easier?

<u>Note:</u>
The purpose of this assignment is to get you familiarized with static analysis, code metrics, code improvements and source code managers. Also it is a first exercise for writing manual tests. All this knowledge is required and useful for workplaces in the domain of software engineering, and once you gather and master this knowledge, you can put this on your CVs.

Also, after you finish this course, you can use this tool on your own programs to improve your coding style, as you will write code in your university studies or as exercise.

Considerations:

Before interpreting the results, it may be good to search on the internet information about Kiviat graphs, metrics and how they are useful. You can also find lots of examples of test analysis, design or test cases. There is no special style of writing tests, code or metric interpretation that is required – it's important that you provide correct and comprehensive information.