CptS422 Software Engineering Principles II (Fall 2019)
Course Project Topics

1. General Description

The objective of this course project is to practice a complete software development process with a clear emphasis on software testing. In order to apply the various testing methodologies and techniques to be learned from this course, your team will first develop the software in a short period of time (two to three weeks). An agile process is recommended. Each team has the full flexity of proposing a topic for this project. Meanwhile, as a reference, a few topics are provided below as samples. These sample topics are identified by the instructor with a clear criteria in mind: they can be developed quickly and will provide a good enough context and basis for practicing testing from unit to system levels with substantial effort needed.

2. Sample Topics

Topic 1: Apartment Renting Web Portal

This is an internet portal dedicated to meet main aspects of the users' needs for apartment renting. It is a forum where Property Owners and Tenants can exchange information, quickly, effectively and inexpensively. It features residential properties for rent, providing information about the properties, rates, locations, and other relevant items. The web portal also provides basic search capabilities based on keywords such as location and prices.

Topic 2: Personal Project Scheduler

The project is to develop a tool for individuals working on various projects, helping keep track of multiple projects and their schedules. The tool allows for adding any number of schedules to be done for each existing project, and each schedule can be tracked from the start till the end of the project cycle. Typical details for each schedule include: project for whom the schedule is to be done, start and end date of the project, total working hours needed and spent already, current status and percentage of work completed, email notifications to project members (e.g., automatically adding a schedule to a member's Google calendar).

Topic 3: An electrical calculator

This software simulates a calculator that is capable of commonly used computations, including but not limited to basic arithmetic calculations. Computing with real numbers need to be supported, allowing users to control the level of precision desired (e.g., precise up to four decimal places). Some advanced computations could also be supported, such as power, root, triangular calculations, and so on.

Topic 4: A text editor

This project will develop a notepad-like text editor for almost all computer literate people . It can be used to create simple text documents. It saves files in plain text format (ANCI text) and supports only a few formatting options . It is used to view or edit text files (files having the extension .txt). Optionally, the software will support editing multiple files (e.g., each on a different tab), and will allow for coloring selected texts.

Topic 5: Extending the Checkstyle plugin for Eclipse to include structural metrics

The project consists of building an Eclipse plugin to compute structural metrics. The project will thus be implemented in Java (so, if you do not prefer working with Java for this course project, this topic may not be a good choice for you). The metrics must be implemented as Checkstyle checks (http://checkstyle.sourceforge.net/) by extending the Eclipse CheckStyle plugin (http://eclipse-cs.sourceforge.net/#!/).

In particular, the extension should support the following structural metrics:

- *Halstead Length* is the sum of the total number of operators and operand [1,2]
- *Halstead Vocabulary* is the sum of the number of unique operators and unique operands [1,2]
- *Halstead Volume* is the program length (N) times the $\log_2$ of the program vocabulary (n) [1,2] : Volume = N $\log_2$ n
- *Halstead Difficulty* is half of the unique operators multiplied by the total number of operands, divided by the number of distinct operators [1,2]
- *Halstead Effort* is the difficulty multiplied by the volume. Effort = DV. Effort was intended as a suggestion for how long code review might take [1,2]
- *Number of comments* [3]
- *Number of lines of comments* [3]
- *Number of looping statements* [3]
- *Number of operators* [3]
- *Number of operands* [3]
- *Number of expressions* [3]
- *Number of variable declarations* [4]

The following references provide explanations of what each of the above metrics means.

[1] M. Halstead. Elements of software science. Elsevier New York, 1977.

[2] Posnett, D., Hindle, A., and Devanbu, P. T. (2011). A simpler model of software readability. In Proceedings of the Working Conference on Mining Software Repositories (MSR), pp. 73--82. URL: http://dl.acm.org/citation.cfm?id=1985454

[3] Buse, R. and Weimer, W. (2010). Learning a metric for code readability. IEEE Transactions on Software Engineering (TSE), vol. 36, pp. 546–558. URL: http://ieeexplore.ieee.org/document/5332232/

[4] Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6), pp. 476-493.

Different from other projects, this one will need you to work based on an existing, real-world (which means greater complexity and engineering effort required) project. According to student feedback in the past, this project can be technically challenging in terms of setting up, compiling, and debugging. Yet it is also a good opportunity to contribute to the open source community---a lot of real-world developers are actually using this plugin everyday. To help reduce your effort, the instructor will provide sufficient support in terms of advice and detailed guiding documents.

3. Self-Proposed Topics

If you would like to work on a topic other than the two suggested above, you may propose your own project topic. However, for self-proposed topics, you need to inform and convince the instructor, before the Deliverable/Milestone 0 deadline, that: (1) the project can be realistically finished up to initial implementation before the Deliverable 1 deadline, and (2) the project will be readily testable (i.e., testing would not be too hard) and there will be a significant amount of work for the testing phase (i.e., the testing would not be trivial).

======

* Acknowledgement: The first four topics are adapted from the original project topic ideas found on https://www.seminarsonly.com/Engineering-Projects/Software/Software-Projects.php.