Assignment 1: Requirements Analysis & Elicitation

CS3213 Foundations of Software Engineering (AY21/22 Sem2)

Submission Deadline: **Tue 18/01/2022, 10 pm**
Discussion: Week 2 and 3

- You must strictly comply with the noted deadline. No late submissions!
- This is an **individual** assignment. Acts of plagiarism are subjected to disciplinary action by the university. Please refer to `https://www.nus.edu.sg/celc/programmes/plagiarism.html` for details on plagiarism and its associated penalties. *Note: all future submissions will the group assignments.*
- Please use appropriate tools to create your solutions (e.g., LibreOffice/Word/LaTeX for textual submissions, or `draw.io` for graphical solutions). Handwritten solutions are accepted only in exceptional cases and if they are very legible.
- Please create **a PDF document** from the solution including a **title sheet** with the exercise sheet number, your name, and matriculation number.
- Please use this scheme as the file name for the PDF document: `assignment_X_YYYYYYYYY.pdf`, where `X` is the exercise number and `YYYYYYYYY` is your matriculation number.
- Please submit this PDF document via LumiNUS. In case of any discrepancies regarding the submission date, the date given in LumiNUS will count.
- There are **2 marks** to be scored for this assignment sheet. The worst score for any assignment sheet is 0 marks.

Overview

The objective of this assignment (and the few following assignments) is to provide you with a feel for what a requirements elicitation process may look when facing the development of a new application or system. This particular assignment prepares you for the requirements elicitation session with project *stakeholders*.

We will kick things off by providing you with a somewhat ambiguous description of the problem, as well as with several pointers and documents to existing systems. Your first task is then to analyze the provided description and systems, and produce questions that will help you clear out some of the misunderstandings, elicit more requirements, or even discard those that may deemed unfeasible. The following (future) assignments will involve the design and development of several requirement specification documents.

Scenario: ”Bob the Tutor”

Bob is a tutor in the programming course for the first-year computer science (CS) students at his university. He had this role in earlier instances of this course, so he knows the challenges ahead of him. It can become quite stressful, also given that this year the number of incoming CS students has increased significantly. As a tutor, Bob does not only have to provide feedback and help the CS students. He also needs to test and grade their submissions. He is unsure how he will cope with so many students and their feedback requests for their programming assignments.

On top of that, Prof. Hopper will be the new lecturer this year for the programming course, and usually, a new lecturer means additions or changes, which cause extra work for Bob. She already informed Bob that this year, the CS students should be able to submit solutions not only in Python but also in the programming languages C and Java. Furthermore, the assignments should no longer be submitted in the form of e-mails to Bob, but via an online submission system. In this way, everybody in the tutor team can keep track of the submissions. Last year, Bob had some issues with his e-mails because some students tried to hijack his account, which resulted in a delicate situation, where Bob was no longer able to check his e-mails, and submissions got lost. Prof. Hopper would like to avoid that in the future. Additionally, she wants to have a weekly PDF report by Bob about the progress in submissions and his grading. Bob will probably start a spreadsheet that contains all this information.

Long story short, Bob now has to prepare additional reference solutions in C and Java, which is not easy for him because he is only an expert in Python. He needs to get familiar with these programming languages to be aware of the typical pitfalls in C and Java to be able to provide meaningful feedback for the students. Usually, Bob takes the submissions, compiles them, and manually runs some test cases. Based on the results, he graded the submissions. He always wanted to automate the build and test process but never found time to figure it out. There are certainly multiple ways to identify the root causes in the programs and also in how to repair the program. But mostly he had no time and just sent the reference solution to the CS students when they had questions for some tutorial exercises. Only on rare occasions, Bob found the time to provide feedback about the fault locations, what the error messages mean, or where to potentially fix the problem. In the past, Bob also offered interactive coding sessions, where he acted as a human debugger who directly gives hints to students as they type their programs. However, this was very time-consuming, and he probably can no longer provide this service to the students. Sometimes, when Bob enjoys his last hot Milo in the late evening, he dreams of a fully automated software system that would help him to cope with all his tutoring tasks. Bob is a senior computer science student, so maybe at some point in the future, he can implement such a system himself...

Collection of available documents and existing systems

- Cheang et al., ”On automated grading of programming assignments in an academic institution”, Computers & Education, Volume 41, Issue 2, 2003.
  `https://doi.org/10.1016/S0360-1315(03)00030-7`
  `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.2552&rep=rep1&type=pdf`

- *Introduction* of Singh et al., ”Automated Feedback Generation for Introductory Programming Assignments”, Programming Language Design and Implementation (PLDI) 2013.
  `https://doi.org/10.1145/2491956.2462195`
  `https://www.microsoft.com/en-us/research/wp-content/uploads/2013/07/pldi13.pdf`

- ”Artemis: Interactive Learning with Individual Feedback” by TUM
  `https://github.com/ls1intum/Artemis`
  `https://arxiv.org/pdf/2110.15134.pdf`

- Online Integrated Development Environment by NUS CS1101s[1]. It uses Source Academy[2], a computer-mediated learning environment for studying the structure and interpretation of computer programs.

- Online Programming Learning Platforms, e.g., `https://www.codecademy.com`

[1] `https://www.comp.nus.edu.sg/~cs1101s/`
[2] `https://sourceacademy.org/playground`

Task 1: Understanding the Problem [1 mark]

Analyze the given scenario and the provided documentation of existing systems. Summarize your findings in a 1 page textual report. Your report should be structured as follows:

1. Problem Description: the basic problem description in your own words
   (*text only*)
2. As-Is-state: the description of the *"as-is"*-state
   (*text, but bullet lists can be included*)
3. To-Be-state: the description of the *"to-be"*-state
   (*text, but bullet lists can be included*)
4. Key Aspects: the identification of at least 3 key aspects that are important for the tutor/lecturer, and 3 key aspects that are important for the student in the context of our scenario
   (*bullet lists are sufficient*)

Grading Comment: To receive the noted full amount of marks, it is necessary to submit a report with the described structure. Each part of the report thereby should contain enough information to deliver the idea of the problem domain. The given scenario description includes specific descriptions about the as-is-state and the requirements for the to-be-state, which you should include in your report. Exceeding the 1-page limit will not directly lead to loss of marks, but you should attempt to be precise in your description.

Task 2: Questions for Requirements Elicitation Session [1 mark]

Understand what is being asked to design and develop. Does it make sense at all? Are some aspects of the requirements to be narrowed, or completely eliminated? Think not only of functional aspects, but non-functional as well. Does providing a solution for one aspect open a gaping hole in another one? Do not forget about security, reliability, performance and ease of use.

Submit **at least 10 questions** for the upcoming requirements elicitation session.

Hint: Check the typical contents/sections in a requirement specification to get ideas about which topics and areas you need to explore with your questions.

Grading Comment: To receive the noted full amount of marks, you need to submit at least 10 questions that relate to the described program domain. Your questions should be diverse and not focus on only one specific section in the requirement specification.