CSCI3100 Software Engineering

Tutorial 1

17:30pm, 1st Feb, 2021 (Mon.) /

17:30pm 3rd Feb 2021 (Wed.)

Please read the questions before the tutorial.

Answer the following problems based on lecture Topics 1 – 3 notes.

1. Software Development (2020 / HW1 / Q2)

Tom is a new software engineer and one day he receives a new task to design an online food-booking system for the business partner of his company. Below are necessary steps to develop a software system, but in a random order:

a. Analyze and specify the feasibility of the goal and requirement of the system;
b. Start coding: implementing classes and structures for the designed modules;
c. Regularly receive feedbacks from customers, and make improvement for the system.
d. Put the food booking system into real use;
e. Design the document of architecture for the food-booking system;
f. Testing the integrated food-booking system;
g. Integrate the modules and classes together to an integrated system;
h. Test the correctness of each class and module separately;
i. Use the design document to divide the overall architecture into small modules, such as the user profile module, the payment module, etc.;
j. Communicate with customers and justify the goal and requirement of the food-booking system;

Questions:

(1) Please re-order the above development steps in the correct order, such that Tom can easily follow and start the development.
(2) If you choose to use the Waterfall model to design the system, please distribute the above steps in (1) to the corresponding phases of the Waterfall model.

2. Software Development Models (2016 / HW1 / Q3)

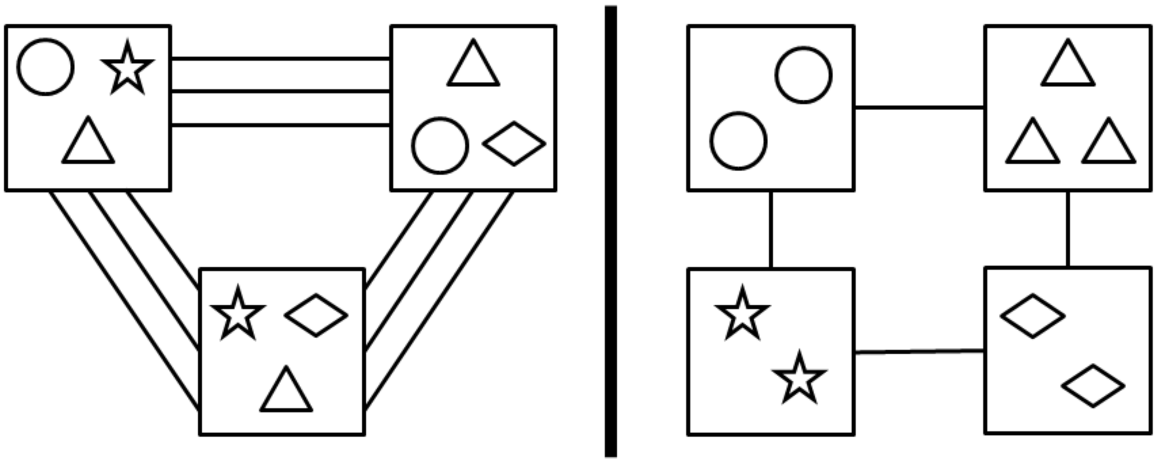
What are the differences between waterfall model and prototype model?

3. Software Development Cost (2017 / HW1 / Q3)

A complex system is usually divided into simpler pieces called modules. Modularity is closely related to separation of concern.

(1) What are the three goals of modularity?

(2) The two diagrams shown below represent two design choices of the same system. They have the same complexity and have the same number of modules. Which one is better and why?



4. **Software Quality (2018 / HW1 / Q4)**

Suppose a university is recruiting software teams to set up a system to manage students' information, including student ID, student name, GPA, average score, credits.

Questions:

(1) A common task of the system is to calculate students' ranking with regard to different aspects. Below is a structure defined by a software team to store the information of a student. Please briefly describe the advantages of such design in terms of *reusability*, which is one of the software qualities we discussed in the lecture.

```
struct student
{
    char StudentName[64];
    int StudentID[8];
    float AverageScore;
    float GPA;
    float Credits;
}
```

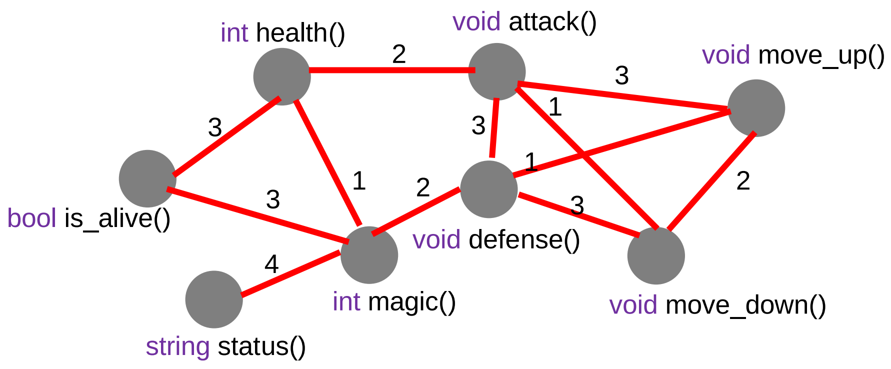
(2) If you are the project manager of a software team that competes for the system, what are the other qualities in your opinion critical for the student information system? Explain the ideas of these qualities in your own words. (Please answer in bullets.)

(3) A software team that competes for the system decides to release their unstable version of their system to users. Please describe the advantages and disadvantages of such strategy. (Please answer in bullets.)

5. **Modularity (2019 / HW1 / Q4)**

A complex software system is usually decomposed into simpler sub-systems called *modules*. Typically, we hope the within-module calls can be more coherent, while the between-module calls can be less coupled. The design for modularity can be considered as a graph partition problem, i.e., we seek to bi-partition G into two disjoint subgraphs G^1 and G^2 by minimizing the between-group calls and maximizing the within-group calls.

Below is an undirected graph representing the calling relations in a game program, in which nodes represent the functions and link weights represent the gain (i.e., communication overhead) of calling.



Questions:

(1) Please mathematically define a good "cut" on the graph into two subgraphs to ensure the minimizing of the between-group calls, and justify your answer. You may denote link weights as w_{ij}, representing the gain (overhead) of calling between function i and function j.

(2) Generally, a good bi-partition of the graph $G(V, E)$ is usually balanced over two subgraphs; for example, in software engineering you may want the system components to be divided in two modules, so that the coherence of two modules should be as even as possible. Give an example of a system where the formula in (1) will not achieve a good composition of two modules.

(3) Give a good bi-partition cut in (2) to the graph above, so that it can achieve the maximizing within-group calls and minimizing between-group calls. Please calculate the resulting gains of within-group calls and between-group calls, and list out the functions included in each module.