21st CSEC – Past Year Paper Solution (2017 – 2018 Semester 2)
CE/CZ 1003 – Advance Software Engineering

1)

a. Heroics encourages extreme risk taking and discourages cooperation among stakeholders in the software-development process. Some managers encourage this behavior when they focus too strongly on can-do attitudes. By elevating can-do attitudes above accurate-and-sometimes-gloomy status reporting, such project managers undercut their ability to take corrective action until the damage has been done. For example, when a developer is uncooperative and refused to work with others as he is overly self-confident, this can result in slow development progress than working as a team. Hence, heroics usually does more harm than good, making heroics a classic mistake in software engineering.

b. The two internal attributes are programme size in lines of code and number of error messages. Generally, the larger the size of code of a component the more complex and error-prone that component is likely to be. Programme size in lines of code allows developers to measure the size of the actual software with the estimated size from using function points. If the size of the actual software is much larger than the expected size, this can imply more bugs in the programme, making the programme less reliable. When there are too many error messages encounted while using or testing the programme, the programme is less reliable. Hence, programme size in lines of code and number of error messages can be used as internal attributes to achieve high reliability of software system.

c. The four factors are organisation and people, client, product, and technology. Agile development method is suitable for project with highly skilled and experienced developers who are able to respond to client's changing requirements and expectations. It can be used in a small or medium sized project where communication between developers is emphasized instead of detailed documentation. We also have to consider the use of technology whether documentation is required. Hence, such strengths and weaknesses had to be kept in mind when selecting Agile development method for the project development.

d. Software metric can be used to predict the product quality. For example, high value of number of overriding operations indicates that the superclass may not be an appropriate parent for the subclass in Object Oriented programme. Thus, the programme is not well-designed.

   Software metrics allow us to identify anomalous component. For example, method fan-in/fan-out can be used to identify private classes are not implemented correctly.

   Software metrics allow software and processes to be quantified. For example, we can quantify the complexity of a software using cyclomatic complexity.

2)

a.

   i. The 5 function points characteristics are inputs, outputs, inquiries, logical files, and interfaces.
      1. get menu info from three shops – 3 interfaces
      2. menu info saved into local database – 3 internal files
      3. browse through the menu – 1 inquiry
      4. place orders by filling up an order page and a payment page – 2 inputs
      5. save the order into the database – 2 internal files (one for order, one for payment)

6. send the payment to a remote system – 1 interface
7. generate a receipt – 1 output
8. check on the details of the order – 1 inquiry
9. provide feedback on the service – 1 input, 1 internal file
10. generate report listing all the orders at the end of each month – 1 output

We assume that only (1) has medium complexity because it involves retrieving menu and sending order to the shop and the rest has low complexity

ii.

Characteristics	**Low Complexity**	**Medium Complexity**	**High Complexity**
# Inputs	$3 \times 3 = 9$	$0 \times 4 = 0$	$0 \times 6 = 0$
# Outputs	$2 \times 4 = 8$	$0 \times 5 = 0$	$0 \times 7 = 0$
# Inquiries	$2 \times 3 = 6$	$0 \times 4 = 0$	$0 \times 6 = 0$
# Internal Files	$6 \times 7 = 42$	$0 \times 10 = 0$	$0 \times 15 = 0$
# External Interfaces	$1 \times 5 = 5$	$3 \times 7 = 14$	$0 \times 10 = 0$
TOTAL:	70	14	0

Total Unadjusted Points = 84

iii. Average on-line data entry: 3
Little data communications: 1
Easy operation: 3 (assume average)
High end-user efficiency: 4
High operational ease: 4
Average complexity processing: 3

Total score = 3 + 1 + 3 + 4 + 4 + 3 = 18
Influence Multiplier = (Total score) * 0.01 + 0.65
= 18 * 0.01 + 0.65
= 0.83
Total Adjusted FPs = Unadjusted FP * Influence multiplier
= 84 * 0.83
= **69.72**

b.

i.

E = 2.94 * EAF * $(KLOC)^{1.0997}$
= 2.94 * (0.71 * 1.40 * 1.30 * 1.54 * 0.82 * 0.83) * $(12)^{1.0997}$
= 61.21

D = 3.67 * $E^{0.3179}$
= 3.67 * $(61.21)^{0.3179}$
= 13.57

Team size = E / D
= 61.21/ 13.57
= 4.50
≈ **5**

ii. $E = 12000 / 640 = 18.75$ PM
$D = 3.0 * (18.75)^{1/3} = 7.96$ month
Team size = E / D
$= 18.75 / 7.96$
$= 2.35$
$\approx$ **3**

3)

a.

i. In the client-server model, developers use a shared single repository. I would propose Concurrent Version System (CVS) and Subversion (SVN) as the two source control software. Both CVS and SVN are open-source centralized source control software. In centralized version control, the server is the master repository which contains all of the code versions. To work on any project, the developer (client) first needs to get the code from the master repository or server to the local machine. Committing a change means merging the changed code into the master repository. Hence, everything is centralized in this model. This is different from distributed version control where every developer has their own server and a copy of the entire history of the code in their local machine.

ii. Function: Track history of the project
While both CVS and SVN allow developers to track the modification of the project, CVS only tracks modification on a file-by-file basis while SVN tracks the whole commit as a new revision.

Function: Commit changes to the code
Both CVS and SVN allow developers to commit changes to the existing version of the code. While SVN allows atomic commit, CVS does not. An atomic commit means the user either apply commit either fully or not at all.

iii. Tracking a commit on a file-by-file basis makes it more difficult for developers to track the modification history of the project. This is because we are often more interested in the version of project rather than version of file.

In CVS, it is possible for the commit to be successful on several files while fail on some other files. This leaves the system in an unfortunate state because part of the commit is missing. As a result, the developer will need to fix the commit as soon as possible before other developers update the project to the broken version. On the other hand, this will not happen in SVN because atomic commit either commit everything or the whole commit fails.

Hence, SVN has two advantages over CVS considering the two functions.

b.

i. This regression testing will aim to test if the function verifyAccount in the system will allow user to login if the user enter correct username and password. The system contains multiple databases including account database and progress database. The username and password should be searched in account database, not progress database. In the account database, an account with username "James" and password "james1234" has been created for testing, which should result in successful login. Initially the developer has implemented the verification function correctly and the whole system works fine. However, while modify the code, the developer changed account database to progress database, introducing a bug to the system as shown in the pseudo code below.

Correct version:

```
function verifyAccount(username, password)
        if match of username and password are found in the account database
                then Allow user to login
        otherwise, unsuccessful user login
```

Wrong version:

```
function verifyAccount(username, password)
        if match of username and password are found in the progress database
                then return true //Allow user to login
        otherwise, return false //unsuccessful user login
```

```
// Testing
Boolean sl = verifyAccount("James", "james1234");
if (sl)
        System.out.println("Pass test.");
else
        System.out.println("Fail test case, test output " + sl + " instead of true");
```

Initially, since the developer used account database to verify the login, the username "James" with password "james1234" can be found in the account database and it will return true which is correct. However, later on when the developer modified the login to use progress database, the function verifyAccount will return false because there is not such username and password in the progress database. By rerunning regression testing, the test case will show that it failed and the developer will be able to do correction.

ii. Agile may benefit more from regression testing. Agile is an incremental development strategy and it enables many changes in requirements, resulting in modifications of code throughout the project. Thus, it is crucial to ensure that modification of existing working code does not introduce bugs and affect the system. However, the requirement is unlikely to change after being finalized in the Waterfall lifecycle.

4)

a.

i. Capability Maturity Model Integration (CMMI)

ii. There are 5 possible ratings known as maturity level in CMMI — Initial, Managed, Defined, Quantitatively Managed, and Optimizing. The level of each organization depends on how well are the processes defined in the organization. At Initial level, processes are unpredictable, poorly controlled and reactive. At Managed level, processes are characterized for projects and is often reactive. At Defined level, processes area characterized for organization and is proactive. At Quantitatively Managed level, processes are measured and controlled. At Optimizing level, organization focus on process improvement.

iii. To achieve a particular level in CMMI, the company needs to define the required key process areas. The company must identify the goals for each key process area. Each of them is organized by common features that address the implementation and contains key practices that describes activities to achieve the goal. For example, in the course project, we are at level 1 Initial. To achieve level 2 Managed (Repeatable), we need to develop all 6 key process areas including software project tracking & oversight and software subcontract management which are missing in the project. In each of the key process area, we define details about the goals and best practices.

b.

i. There are 4 strategies to evolve such systems.
   1. Scraping the system completely and modify business processes so that it is no longer required
   2. Continue maintaining the system
   3. Transform the system with re-engineering to improve its maintainability
   4. Replacing the system with a newer system.

ii. We can determine which strategy to use based on the business value and quality of the existing system.
Low business value, low quality system: Scrap system completely
Low business value, high quality system: Continue maintaining the system, scrap system
High business value, low quality system: re-engineering, replace the system
High business value, high quality system: Continue maintaining the system

Solver: Chulpaibul Jiraporn (chul0004@e.ntu.edu.sg)