Assignment 4—Solution

Handout: **Monday, November 15th, 2021**
Due: **23:59, Thursday, November 25th, 2021**

1. Software Maintenance (12 marks)

As introduced in lecture one, there are four main types of maintenance in the modern view of software maintenance. Briefly describe the three main types and explain why it is sometimes difficult to distinguish between them. (≤200 words)

The three main types of software maintenance are:

1. Corrective maintenance. The changes made to the system are to repair reported faults which may be program bugs or specification errors or omissions.

2. Adaptive maintenance. Changing the software to adapt it to changes in its environment, e.g., changes to other software systems.

3. Perfective maintenance. This involves adding new functionality or features to the system.

4. Preventive maintenance. The changes are made to correct undiscovered faults.

They are sometimes difficult to distinguish because the same set of changes may cover all three types of maintenance. For example, a reported fault in the system may be repaired by upgrading some other software and then adapting the system to use this new version (corrective + adaptive). The new software may have additional functionality and as part of the adaptive maintenance, new features may be added to take advantage of this.

2. Software Evolution (26 marks)

Suppose that the requirements for the Jungle Game in your project need to be changed to support an “auto” mode for players. A player can enter the auto mode by inputting “auto” when it is his/her turn to make a move, and once entering the auto mode, the player cannot exit the mode. When a player is in the auto mode, the computer will take over and make random decisions for the player in each turn until the game is over.

Please briefly explain what important changes would be necessary for your current game design to support the new requirements. (≤300 words)

The answer is accepted as long as it makes sense.

3. Software Reuse (12 marks)

Give at least two benefits of and two problems with software reuse. (≤200 words)

Benefits

- *Accelerated development. Bringing a system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time may be reduced.*
- *Effective use of specialists. Instead of doing the same work over and over again, application specialists can develop reusable software that encapsulates their knowledge.*
- *Increased dependability. Reused software, which has been tried and tested in working systems, should be more dependable than new software. Its design and implementation faults should have been found and fixed.*
- *Lower development costs. Development costs are proportional to the size of the software being developed. Reusing software means that fewer lines of code have to be written.*
- *Reduced process risk. The cost of existing software is already known, while the costs of development are always a matter of judgment. This is an important factor for project management because it reduces the margin of error in project cost estimation.*
- *Standards compliance. Some standards, such as user interface standards, can be implemented as a set of reusable components.*

Problems

- *Creating, maintaining, and using a component library. Populating a reusable component library and ensuring the software developers can use this library can be expensive. Development processes have to be adapted to ensure that the library is used.*
- *Finding, understanding, and adapting reusable components. Software components have to be discovered in a library, understood, and sometimes adapted to work in a new environment. Engineers must be reasonably confident of finding a component in the library before they include a component search as part of their normal development process.*
- *Increased maintenance costs. If the source code of a reused software system or component is not available, then maintenance costs may be higher because the reused elements of the system may become incompatible with changes made to the system.*
- *Lack of tool support. Some software tools do not support development with reuse. It may be difficult or impossible to integrate these tools with a component library system.*
- *Not-invented-here syndrome. Some software engineers prefer to rewrite components because they believe they can improve on them. This is partly to do with trust and partly to do with the fact that writing original software is seen as more challenging than reusing other people's software.*

How to hand in:

Submit your typed, instead of handwritten, answers in a single PDF file on Blackboard.