Singapore Institute of Management
BSc in Computing & Information Systems
CIS 210 Tutorial 5 Suggested Solution

[Note: The answers are suggested solutions for your reference. They are not to be treated as model answers but rather a guide on how you might answer the questions]

1. What is coupling? Why is common, data & content coupling deemed as good or bad? Justify your answer.

Answer :

Coupling measures the "selectiveness" of a module with respect to what kind of dependencies and how many it has with other modules. Level of coupling depends on

- *What data passes through the interface*
- *Where does the data come from or is stored*

Common coupling – 2 or more modules read or write to a common data area. This situation is increasingly rare.

Data coupling – complete piece of data flows between the 2 modules typically a super- and a subordinate.

Content coupling – module reads or writes to another module's data area or branches into its body.

Both Common and content are deemed as bad coupling. For these types of coupling, the modules are not very "selective" in information sharing.

For common coupling, data in one are can be changed by any one module sharing the data area. This makes it difficult to determine which module will change the data at which point in time during program execution. Debugging and maintenance is difficult.

For Content coupling, one program/module can write to the data area and branch even into another program /module. This is dangerous, as this means that the target module's execution/data becomes uncertain at any point in time during execution. Again debugging and maintenance will be difficult.

For data coupling, a complete piece of data flows from 1 module to another. The number of modules and their actions on the data can be determined during static and dynamic execution. This makes it easier to debug and maintain programs.

2. Discuss what Cohesion means in Software Engineering. Is Logical and Communicational cohesion good or bad? Why?

Answer :

Cohesion – a module is more cohesive if it provides less functionality than it provides beyond the one it is mainly responsible for. A module should do just one well-defined task.

Logical Cohesion is that a relationship exists between the component tasks as the tasks belong to common category. For e.g. if module is made up of 2 component tasks :

Print statement
Print report

Communicational cohesion is that tasks in the module operate on a common data structure. An e.g. will be

Update name
Print date-of-birth

Logical is bad cohesion as the tasks are grouped only because they are of the same category. They are not working to provide one coordinated service to the calling module. In the above e.g. the tasks are all printing tasks. The statement printing can be a separated from the report printing task with no side effects. Thus, there is no need to put these 2 tasks in 1 module.

Communicational cohesion is good as the tasks works on a common data structure. Here all the tasks work on the personal details of a person/client. The tasks are grouped together for the purpose of acting on the details of the client.

3. Why is prototyping useful in the UID process? Give an example to support your answer. What other methods can be used in the UID?

Answer :

User Interface Design (UID) is to define the user interface during the software engineering development process. Prototyping is useful in UID as a prototype is developed during the prototyping process to show the customer. Feedback on whether the UID provided in the prototype is suitable can then be gathered and with prototyping process being an iterative one, the appropriate UI can then be developed to suit the customer's requirements. Thus prototyping is useful in the UID process.

For e.g. if a customer does not know what kind of user interface is required, a UI designer can then design prototype of the application based on the rough requirements gathered. The customer can then view the prototype and determine if the UI is suitable. Comments and feedback can be gathered from the customer during the evaluation task within the prototyping process. Necessary changes can then be made to the prototype based on the feedback. The revised UI on the prototype can be shown again to the customer. The process is repeated to design the UI until the customer accepts the UI. In addition, usability tests can be conducted with the prototype to determine if the UI is easy to use and the changes required. Thus prototyping can be very useful during the UID process.

Other methods useful in the UID includes : questionnaires, ratings, measured interactions, etc.

4. What are the major differences between function-oriented design and object-oriented design? Give instances of when you would use which design and justify with reasons.

Answer :

For Functional design, system is designed from a functional viewpoint, starting with a high-level view and progressively refining this into a more detailed design. The system state is centralized and shared between the functions operating on that state.

For Object-oriented design, system is viewed as a collection of objects rather than functions. Object-oriented design is based on data of information hiding. Unlike functional design which is based more on data sharing. System state is decentralized and each object manages its own state information. Objects have their own attributes defining their state and operations acting on the attributes.

An e.g. is a software systems that is part of a modern civil aircraft. Natural high-level overview of this software is a set of sub-systems or objects. At this abstract design level, an OO approach is natural.

When the system is examined in detail, its natural description is a set of interacting functions rather than objects e.g. displaytrack(radar subsystem), compensateforwindspeed (navigation subsystem) etc. This functional overview may be taken by the requirements definition. (note that difficult t validate the system as no simple correspondence between design and requirements)

Detailed design stage, an oo view may again become the natural way to view the system. E.g. of objects at this stage include "the_engine_status", "the aircraft_position", etc. OO approach at lower levels of system design is likely to be effective.

The 2 methods of design are complementary rather than opposing. OO approach is natural at highest and lowest levels of system design.

5. Describe the different stages of UI development over the last few decades. What principles would you use in developing UI for any software projects that you are involved in?

Answer :

1st generation UI – command-line based interfaces where the communication is entirely textual. System prompts user and user types commands, system prints response and prompts user again and so on. e.g. the MS-DOS Operating System

2nd Generation UI – menu-driven interfaces. Comms is still textual. But here the prompt is a list of valid next actions and the user needs only to select an option from the restricted set of valid alternatives. Less error-prone. E.g. is Gopher.

3rd generation UI – WIMP (windows, icons, menus, pointing devices). Comms is mainly visual. Users are offered multiple contexts and can switch between them. Multiple tasks are simultaneously available via pull-down menus. Icons, scroll-bars etc reduce need for typing. E.g. Microsoft windows OS.

Principles of UI development :

- *User familiarity – interface should use terms and concepts which are drawn from the experience of the anticipated class of user*
- *Consistency – UI should be consistent in that comparable operations should be activated in the same way*
- *Minimal surprise – Users should never be surprised by the behaviour of a system*
- *Recoverability – the UI should include mechanisms to allow users to recover from their errors*
- *User guidance – interface should incorporate some form of context-sensitive user guidance and assistance*

6. Suggest a structural model for each of the following systems :
   a) An automatic ticket issuing system used by passengers at the train stations
   b) A computer-controlled video conferencing system which allows video, audio and computer data to be shared by several participants simultaneously
   c) A robot floor cleaner which is intended to clean clear spaces such as corridor. Cleaner must be able to detect the walls and obstructions if any.

   Justify your answers.

Answer :

a) Repository model because

- *Shared data is held in central database to be shared by the systems at the various stations*
- *Each system at the train station can have its own database*
- *Useful for sharing large amounts of data among the different ticket sharing systems installed in the various train stations*
- *Each subsystem in each train station need not be concerned how data is used by other sub-systems*
- *Activities such as backup, security access etc are centralized. Responsibility of repository manager.*

b) Client-Server model because

- *Have a set of servers (such as the video, audio and computer components) which offer services to the other sub-systems.*
- *The client components can call on the services of these server components.*
- *Have limited server components whose names can be made easily available to the clients for access of services through remote procedure calls.*
- *Network allows the clients to easily access the services.*

c) Abstract machine model because

- *Organise system into a series of layers each of which provides a set of services*
- *1 layer defines the robot floor cleaner*
- *the next bottom layer is the detection layer*
- *The next bottom layer is the database layer*
- *The final layer is the OS*
- *This architecture is changeable and portable. Any one layer can be replaced by another layer. When 1 layer is changed only the adjacent layers are affected. Layered systems localize machine dependencies in inner layers and they can be ported easily.*