Theory Assignment-4

1	Mishal Shah	16CO125	shahmishal1998@gmail.com	9725801399
2	Samyak Jain	16CO254	16co254.samyak@nitk.edu.in	9818099551

What, Why, How?

In your own work(mini project), How will you use these in our project

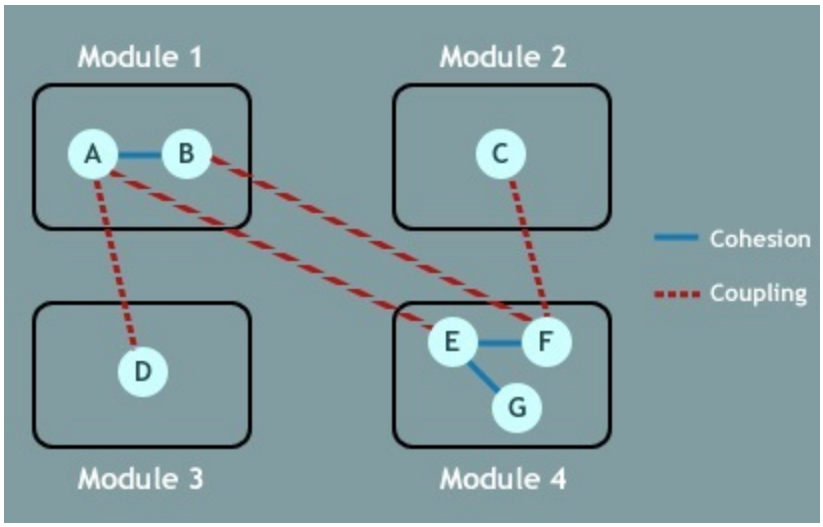
1. Coupling & Cohesion
2. Conceptual, Technical Design, Exception Handling, Fault Tolerance

Coupling and Cohesion

A good software needs to be divided into several modules based on characteristics. These modules contain the information based on the the tasks we plan to achieve. They are working all for the single entity but will have to refer to each other for the whole system to work. The measures by which the quality of the design of modules and their interaction between them is measured is known as coupling and cohesion.

Cohesion is a measure that defines the degree of dependability within the elements/components of module.

Coupling is a measure that defines the level of inter-dependability among the different modules of the program.



For a good system, there should be more cohesion and less coupling.

Why is coupling and cohesion required?

Coupling and cohesion is required to make the entire big and complex system into different modules that serve independent functions and are loosely coupled with each other. Different components of the module may be strongly cohesive in nature to give better results. The advantages of modularization is:

- Parallel Development - Modularization ensures that loosely coupled modules can be developed in parallel
- Debugging: Modularization makes it easy to debug the code. In a loosely coupled design the number of interfaces and the number of parameters/data passed are low, thereby ensuring easy debugging of code.
- Reusability: if one module (say a Class) has to be changed then only that can be changed and others can be reused, thus a loosely coupled system is advantageous.
- Maintainability: The independence enforced by loose coupling improves maintainability of the system i.e. if there is a future requirement then it can easily be addressed looking at the module (say a Class) and the component that requires the change.
- Robustness - Highly Cohesive modules loosely coupled together are a measure of software robustness (i.e. quality) and is a highly desirable feature

How coupling and cohesion is done in our project?

In our mini project several modules exists, they are:

1. Patient module: In this module components handling patient registration, patient consultation request will be there. These components need to be highly cohesive as they are strongly dependent on each other.
2. Doctor module: In this module components handling doctor details and prescription generated by doctor. Again these components need to be strongly bonded together in order to make this module work hence high cohesion.
3. Chemist and Lab module: These modules have lab and chemist details and the services provided by them.
4. Admin module: These module will have components which can add doctors, chemists, labs, get a summary of the patient details, appointments made.

If one looks all the modules then one can find all of them can do their work independently but still needs to be loosely coupled with each other. E.g. a doctor can provide consultation only if there is a consultation request by a patient and similarly a chemist can deliver medicines only if a doctor generates a prescription.
The reason why our system will have loose coupling and tight cohesion is: The template system will know nothing about Web requests, the database layer will know nothing about data display and the view system doesn't care which template system a programmer will use.

An example for bad loose coupled system is your URL being coupled to underlying code. We will be using MVC architecture for it uses loose coupling and tight cohesion.

Conceptual, Technical Design, Exception Handling, Fault Tolerance

Conceptual Design: Its a design that tells the customer exactly what the system will do. (The What of the solution). It is written in customer's language so that they can understand the system easily. It explains the observable external characteristics of the system. It does not contain any technical jargon. It clearly describes the functions of the system. It is independent of the implementation of the system. It is linked to the requirements documents i.e. SRS

Technical Design: It is the design that allows developers to understand the actual hardware and software needed to solve the customer's problem. (The How of the solution). It describes of major hardware components and their functions. It shows hierarchy (organization) and functions of the software components. It shows data structures and data flow. It also shows interfaces of the system.

Exception Handling: Exceptions are anomalous or exceptional conditions requiring special processing. It often changes the normal flow of program execution. So we need to build our system in such a that it handles these situations. Exceptions include failure to provide a service, providing the wrong service or data or corrupting data. Exception Handling means either to retry or correct or report the exception.

Fault Tolerance: It is the property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components. The ability of maintaining functionality when portions of a system break down is referred to as graceful degradation. A fault-tolerant design enables a system to continue its intended operation, possibly at a reduced level, rather than failing completely, when some part of the system fails.

Why Conceptual design, Technical Design, Exception handling and fault tolerance?

Conceptual design: It is needed so that customers can get to know what the system will look like. After telling all their requirements this design will let them know of "what is the solution" to their problem E.g. If we want to build a house we tell all the requirements to the builder, now conceptual design is nothing but a blueprint of the house that shows where each room is and how is it organized.

Technical Design: It is needed by the system developers in order to get to know how the system should be developed. It clearly outlines all the software and hardware requirements of the system which developers use to develop the system. E.g. in construction of the house not only simple blueprint but a detailed design of each component of the house is given along with all technicalities to construct the house.

Exception Handling: It is needed so that unforseen or unanticipated events in the system can be handled and it does not lead to failure instead it is either reported, or corrected or retried. E.g. If in our computation a divide by 0 occurs then the system should trap it before execution and report it.

Fault Tolerance: It is required so that system is capable to handle faults and it does not lead to failure of the system.

How Conceptual Design, Technical Design, Exception Handling, Fault tolerance in our project?

Conceptual Design: For our mini-project we will develop a very simple conceptual Design module by showing the functional components ("What") and the interaction between them.

Technical Design: The Detailed System Design Document for our mini-project shall detailed technical design for our project. We are intending to follow the UML approach to system design and our design shall include - Use Case Diagram, Class Diagrams, Activity Diagram, Sequence and Collaboration Diagrams, State Machine Diagram, Component Diagram and Deployment Diagrams.

Exception Handling: There can be many exceptions that our project will have to cater to. For instance, our project is intended to provide services in certain areas only (Zone/ Pincodes). Now a patient, trying to register from outside this area should automatically report it and handle this exception ("Error - This service is not currently available"). Another instance is when a Doctor tries to login whose details have not been set-up in the System. The System will trap this as an Exception and report an Error "You are not authorized to use the System - Contact System Administrator".

Fault Tolerance: In our project, we have made provisions for the system to roll-back to previous date in case of a database crash/ accidental corruption of the run-time libraries. This will be ensured by restoring from previous date' back-up.