gradeup

Software Design Part

Software Design:

Content:

Abstraction: It is one of the concept of software engineering. It is about hiding complexity in building many parts of your application.

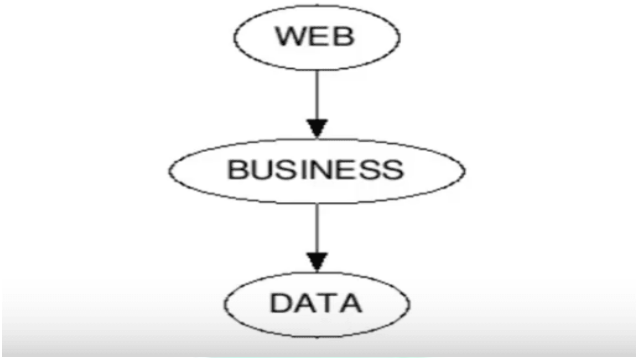
An Example of Abstraction:

Consider a real-world analogy. User want to ride a motor bike. All user need to start the motor bike is to put the key on, push the start button, and use the accelerator. While riding, user may need to use the brake as well.

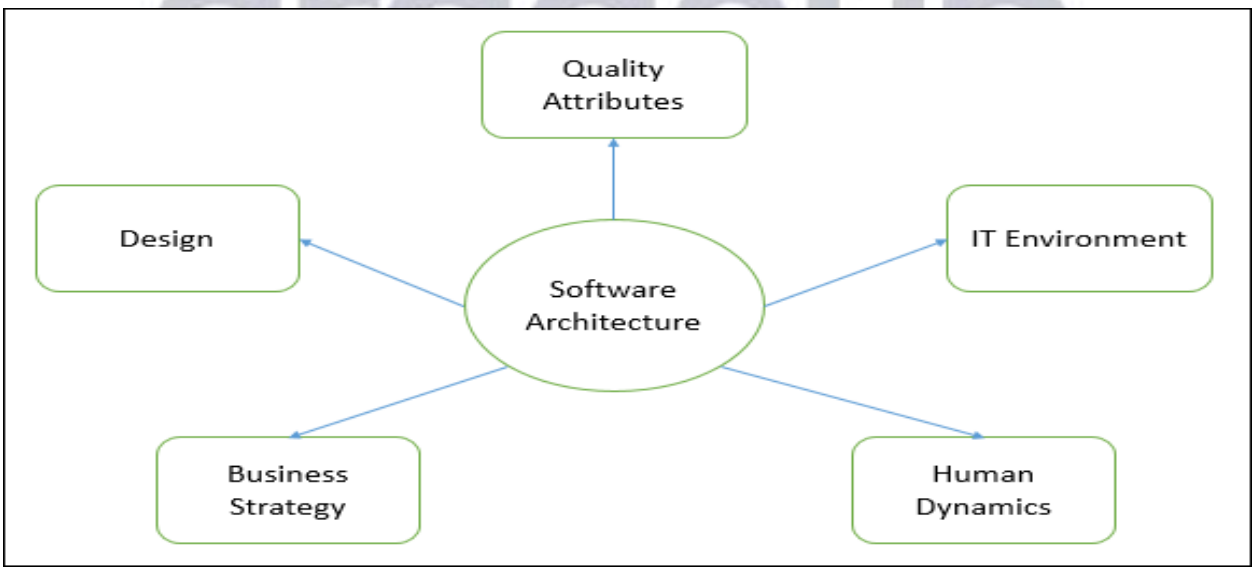

Gradeup UGC NET **Super Subscription**
Access to all Structured Courses & Test Series

You are not really concerned about how the engine, accelerator and brake are working during the ride. All that is abstract to user and you are not concerned with it (unless you are mechanic).

Whenever we build projects, wedo in layers. Here is the layered architecture of a simple web application:



Architecture: It is a system describes its major components, their relationships (structures), and how they react with each other. It includes many factors such as Business strategy, quality attributes, human dynamics.



Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

It serves as a blueprint for a system. It provides an abstraction to control the system complexity and develop a communication and coordination mechanism among components.

It defines a solution to meet all the technical and operational requirements, while consuming the common quality attributes like performance and security.

It involves a group of significant decisions about the organization related to software development and one of these decisions can have a considerable impact on quality, maintainability, performance, and the success of the last product.

These decisions comprise of :

- Selected elements and their interfaces by which system is composed.
- Behavior specified in groups among those elements.
- Composition of those structural and behavioral elements in large subsystem.
- It align with business objectives.
- It styles guide organization.

The goal of this is to identify requirements that affect the structure of the project. Architecture decreases the business risks associated with building a technical solution and builds a path between business and technical requirements.

Some of goals are:

- Expose the structure, but hide its implementation details.
- Realize all the use-case and scenarios.
- Try to address the requirements of some stakeholders.
- Handle functional and quality requirements.

Limitations:

- Less of tools and standardized ways to present architecture.
- Less of analysis methods to predict whether architecture will result in an implementation that meets the requirements.
- Less of awareness of the importance of architectural design to software development.


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

- Less understanding the role of software architect and bad communication among stakeholders.

Patterns: It represents the good practices used by experienced object-oriented software developers. They are solutions to basic problems that software developers faced during software development. These solutions were taken and error by numerous software developers over a quiteshort period of time.

Modularization:

It is a technique to divide a software into couple of discrete and individual modules, which are expected to be capable of carrying out task independently. These modules work as basic builds for the entire software. Users tend to design modules such that they can be executed or compiled separately and independently.

It follows the rules of 'divide and conquer' problem-solving method this is because there are many other advantages attached with the modular design of a software.

Advantage of modularization are:

- Smaller components are easier to maintain
- Application can be divided based on functional aspects
- Required level of abstraction can be brought in the program
- High cohesion can be re-used again
- Concurrent execution can be made possible
- Desired from security aspect


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

For example, the Linux kernel is modular. Various computer program languages refer to their libraries as "modules." Including a module in your program adds new capabilities by granting access to pre-written code objects.

Functional independence: It means that when a module focuses on a single task, it should be accomplish it with very few interaction with other modules. It is essential for good software design.

Advantage of functional independence:

- **Error isolation:**

When a module is functionally independent then it performs most of its task independently without interacting with other modules much. It reduces the error getting propagated to other modules. It helps easily isolating and tracing the error.

- **Module reusability:**
  It performs well defined and specific task. So it becomes easy to reuse such modules different program require same functionality.

- **Understandability:**

It is less complex so easy to understand. Such modules are less interacted with other modules so can be understand in isolation.

Cohesion: It is a measure of degree to which the elements of the module are functionally similar. A degree to which all elements directed towards performing a single task are stored in the component. It is the internal glue that keeps the module together. A software design will have high of cohesion.

Types of Cohesion:

- **Functional Cohesion**: Each required element for a single computation is stored in the component. It performs the task and functions. It is an ideal situation. Example : read transaction record, cosine angle computation, seat assignment to an airline passenger etc.


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

- **Sequential Cohesion**: An element results some data that becomes the input for other element, i.e., data flow between the parts. It occurs naturally in programming languages. Example: cross validate record and formatting of module, raw records usage, formatting of raw records, cross validation of fields in raw records.

- **Communicational Cohesion**: Two elements operate on the same input data or distribute towards the same output data. Example- modify record into the database and send it to the printer.

- **Procedural Cohesion**: It ensure the order of execution. Actions are still connected and unlikely to be reusable. Ex- calculate student GPA, print student record. Example : read, write, edit of the module, record use out, writing out the record, reading the record, zero etc.

- **Temporal Cohesion**: The elements are similar by their timing involved. It isconnected with all the tasks must be executed in the same time-span. It contains the code for initializing the parts of the system. Example: A function is called after catching an exception which closes open files, creates an error log.

- **Logical Cohesion**: All elements are logically similar and not functionally. Ex- It reads inputs from tape, disk, and network. The code for these functions is in the same component. They are similar, but the functions are different. Example: a group of print functions generating different output reports are arranged into a single module.

- **Coincidental Cohesion**: The elements are not related. The elements have no abstract relationship other than location in source code. It is the


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

worst form of cohesion. Ex- print next line and reverse the characters of a string in a single component, a “Utilities” class.

- **Coupling:** It is the measure degree of interdependence between the modules. Asoftware will have low coupling.

Types of Coupling:

- **Data Coupling**: Tthe dependence between the modules is based on the strategy that they interact by passing only data, then the modules are said to be data coupled. In data coupling, the components are freely to each other and communicating through data. Module communications don’t contain tramp data. Example-customer billing system. Example : illustrated as a module which retrieves customer address using customer id.

- **Stamp Coupling:** In this, the complete data structure is passed from one module to another module. It involves tramp data. It may be mandatory due to efficiency factors- this choice made by the insightful designer, not a lazy programmer.

- **Control Coupling**: If the modules meet by passing control information, then they are to be control coupled. It can be worst parameters showed completely different behavior and good if parameters allow factoring and reuse of functionality. Example- sort function that takes comparison function as an argument. Example : a **control** flag, a comparison function passed to a sort algorithm.

- **External Coupling:** The modules depend on another module, outside to the software being establish or to a particular type of hardware. Ex- protocol, external file, device format, etc.


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

- **Common Coupling**: The modules have shared data such as global data structures.The changes in data mean tracing back to all modules which access that data to evaluate the effect of the change. Disadvantages are likely to difficult in reusing modules, reduced ability to control data accesses. Example : global information status regarding an operation, with the multiple modules reading and writing to that location.

- **Content Coupling**: In this ,module can modify the data of another module or control flow is passed from one module to the other module. This is the bad form of coupling and should avoided.

Difference Between Cohesion and coupling:

Cohesion	Coupling
It is the concept of intra module.	It is the concept of inter module.
It represents the relationship within module.	It represents the relationships between modules.
Increasing in cohesion is good for software.	Increasing in coupling is avoided for software.
It represents the functional strength of modules.	It represents the independence among modules.
Highly cohesive gives the best software.	Loosely coupling gives the best software.

Object-Oriented Design:

The system is showed as a collection of objects. The state is distributed among the entities, and each object handles its data. For example, in a Library Automation Software, every representative may be a separate object with its information and functions to operate on these data. The task state for one purpose cannot refer or change data of other objects. They have their internal data which represent their state.


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

Data Design:It is the first design activity, which results in less complex, and efficient program structure. The data objects, attributes, and relationships picture in entity relationship diagrams and the application stored in data list provide a base for data design activity. During the process, data types are specified along with the integrity rules required for the data.

Architectural Design: **It is a process for knowing the sub-systems making up a system and the framework for sub-system control and communication. The output of this design process is detailed of the software architecture. It is a stage of system design process.**

An architectural design performs the following functions :

1. It evaluates all level designs.
2. It develops and documents level design for the outside and inside interfaces.
3. It develops preliminary versions of user documentation.
4. It defines and documents preliminary test requirements and the schedule for software integration.
5. Information of the application domain for software to be developed.
6. Using data-flow diagrams.

User Interface Design: It is the front-end application view to which user interacts to use the software. User can control the software as well as hardware user interface. It is found at each place where digital technology exists, mobile phones etc.

It is a part of software and is designed in such a way that it is expected to provide the user insight of the software. It provides fundamental platform for human-computer interaction.

It may be graphical, text-based, audio-video based, depending upon the hardware and software combination. It can be combination of both.

The software becomes more famous if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to understand
- Consistent on all interfacing screen


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

gradeup SUPER

Gradeup UGC NET
Super Superscription

Features:

1. 7+ Structured Courses for UGC NET Exam
2. 200+ Mock Tests for UGC NET & MHSET Exams
3. Separate Batches in Hindi & English
4. Mock Tests are available in Hindi & English
5. Available on Mobile & Desktop

Gradeup Super Subscription, Enroll Now

gradeup.co