Assignment-1

1. Give three examples of seven categories of different software application domain ?

1. System Software :- Computer Language Translators, Disk Formatting, Antivirus.

2. Application Software :- Adobe photoshop, VLC media Player, Microsoft Office

3. Engineering & Scientific Software :- AUTOCAD, PSPICE, ORCAD

4. Embedded Software :- GPS devices, factory robots, modern smartwatches.

5. Product - line Software :- Word Processing, Spreadsheets, Computer Graphics

6. Web Applications :- Online forms, shopping carts, file conversion.

7. Artificial Intelligence Software :- Cortana, Siri, Google Assistant

2. Define legacy Software and give three examples.

Ans:- Legacy software is a software that has been around a long time and still fulfills a business need. It is mission critical and tied to a particular version of an operating system or hardware model (vendor lock-in) that has gone end-of-life. Generally the lifespan of the hardware is shorter than that of the software. As time goes on, the hardware gets harder to maintain, but is kept because it is installed and working and has proven too complex and/or expensive to replace.

With time passes, legacy system often evolve for one or more of the following reasons:-

- The software must be adapted to meet the needs of new computing environment or technology.
- The software must be enhanced to implement new business requirement.
- The software must be extended to make it interoperable with other more modern system.
- The software must be re architectured to make it viable within a network environment.

Examples:-

- A still in-use voicemail
- Electronic microscopes
- Software that keeps score in bowling alleys.

3. Design software engineering layers for any 3 software qualities.

Example1:-
Process:- Project should end in given time, correct feasibility study.
Method:- Identification of resources.
Tools:- Timeline charts etc.

Example 2:-
Process:- It should be feasible with trending technologies and upcoming technologies.
methods:- Identification of current technology trends.
tools:- Decision trees etc.

4. Differentiate between Umbrella activities and framework activities with example.

	Framework Activities	Umbrella Activities
1.	Generic activities that are applicable to all software projects regardless of their size & complexity.	Complementary activities applied throughout a software project and help manage and control progress, quality change & risk.
2.	Includes communication, planning, modeling, construction, and deployment.	Includes project tracking and control, risk management, software quality assurance, technical reviews, configuration management (CM) etc

5. List 7 principles of software engineering?

1. The Reason It All exists:-
All decisions should be made with keeping in mind "to provide value to all its users". Before doing anything we should think if it provides or adds real value to the system.

2. KISS (Keep it Simple, Stupid):- All the design should be as simple as possible but no simpler. This facilitates having a more easily understood and easily maintained system.

3. Maintain the vision:-
A clear vision is essential to the suuess of a software project. Without one, a project almost unfailingly ends up being "of two minds". Compromising the architectural vision of a software system weakens. and will eventually breake even the well designed systems.

4. What you Produce, Others will consume.
Always specify, design and implement knowing someone else will have to understand what you are doing. Code with concern to those that must maintain and extend the system.

5. Be open to future:-
A system with a long lifetime has more value. Softwdee should be able to get reused in the future. Systems must be ready to adapt the changes.

6. Plan Ahead for reuse:-
Reuse saves time and effort. Achieving a high level of reuse is arguable the hardest goal to accomplish. in developing a software system. To leverage the reuse possibilities that OO programming provides requires forethough & planning

7. Think :-
Placing clear, complete thought before action almost always produces better results. When you think about something you tend to do it right. You also gain knowledge about how to do it.

6. List any 5 software myths and corresponding realities!

Myth:- If I decide to outsource the software project to a third party, I can just relax and let them build it.

Reality:- If an organisation does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.

Myth:- A general statement of objectives is sufficient to begin writing programs - we can fill in the details later.

Reality:- Although a comprehensive and stable statement of requirement is not always possible, an ambiguous "statement of objectives" is a recipe for disaster. Unambiguous requirements are developed only through effective & continuous communication between customer and developer.

Myth:- Once we write the program and get it to work, our job is done.

Reality:- Someone once said that "the sooner you begin 'writing code', the longer it'll take you to get done". Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the 1st time.

Myth:- Until I get the program "running" : I have no way of assessing its quality.

Reality:- One of the most effective software quality assurance mechanisms can be applied from the inception of a project - the technical review. Software reviews are a 'quality filter" that have been found to be more effective than testing for finding certain classes of software defects.

Myth:- The only deliverable work product of a successful project is the working program.

Reality:- A working program is only one part of a software configuration that includes many elements. A variety of work products (e.g:- models, documents, plans) provide a foundation for successful engineering and more important guidance for software support.

7 Differentiate between hardware and software failure

	Hardware failure	Software failure
1.	Hardware failure are mostly physical faults.	Software failure are due design failures, which are tough to visualize, classify, detect & correct.
2.	Hardware components generally fail due to wear & tear.	Software components fail due to bugs.

In hardware design faults may also exist but physical faults generally dominate.	In software we can simply find a exact corresponding counter part for "manufacturing" as the hardware manufacturing process, if the simple action of uploading software modules into place does not occur. Therefore the quality of the software will not change once it is uploaded into storage & start running
Hardware exhibits the failure features shown here.	Software reliability does not show the same feature similar as hardware.

Assignment - 2.

1. Answer the following with respect to your OOAD case study.

a) List 5 umbrella activities
- Risk management
- Software quality assurance (SQA)
- Software Configuration management (SCM)
- Measurement
- Formal technical Reviews.

b) List 5 framework activities
- Communication
- Planning
- Modeling
- Construction.
- Deployment

c) Give 5 task sets for each of the above framework activities.

1) Communication
- Make a list of stakeholders
- Invite all stake holders to an informal meeting
- Ask each stakeholder to make a list of feat and functions required.
- Discuss the requirements and build a final list.
- Note areas of uncertainity.

2) Planning :-
- Discuss the technical related task like the software platform requirement, hardware require ments etc.
- Discuss the work schedule
- Identify the risks involved in the process

- Identify the work products
- The resources that will be required

3. Modeling :-
- An appropriate model is selected that is to be following for the software development
- The requirements are modeled.
- Algorithms and flowcharts are designed
- Different use case scenarios are identified.

4. Construction :-
- Generating code for the Software.
- Documentation using comments.
- Testing each unit for bugs.
- Integrating units together.
- Testing the integrated units together

5. Deployment
- Delivering Software to the stakeholders
- Giving user manuals for operating the software
- Providing maintenance.
- Acquiring feedback for customer.
- Providing servicing activities in case of bugs

e) Explain how would you fit different development models in your case study. Which model is best suitable for your case study give reasons.

Ans:- Case Study :- Online Medical Care System

- Spiral Model will be best for Online Medical care System.
- It will also involve risk managment.
- It is good for large projects.
- As the requirements for the various facilities changes or more functions are required it will be flexible towards these changes.
- Customers will be able to see the product at an early stage and if there is any problem e.g some medical stores are not mentioned, specialised doctors are not there etc can be solved at any early stage.

2. Give live example and elaborate one of the given example for following process models:

a) Waterfall Model:-

Water fall model was used to develop enterprise applications like Customer Relationship Management (CRM) systems, Human Resource Management Systems (HRMS), supply chain management systems, Inventory management system, Point of Sales (POS) system for Retail Chains etc.

- Development of Department of Defense (DOD), military aircraft programs followed waterfall models in many organisations

This is because of the strict standards and requirements that have to be followed. In such industries, the requirements are well known in advance and contracts are very specific about the deliverable of project. DOD agencies typically considered waterfall model to be compatible with acquisition process and rigorous oversight process required by the govt.

b) V- Model :-
Medical Devices industry, Tax related software, Small internal projects

Tax related software with respect to the calculations, which is based on laws and regulations. In these cases tax payer feedback is of little use. In this case of developing regulation based software from scratch, more detailed requirements are required to precisely explain the law, database logic and initial report construction because most developers do not have tax or other regulatory experience

c) Incremental model :-
ms-Paint, Spotify.

Spotify uses incremental model. Each time the team releases a new increment of the application and having feedback from customers. By doing this the team was

able to deliver what customers really wanted as opposed to what they thought the customers would want.

d) Prototyping Model:-
E-commerce websites, Command driven Systems

Ebay:- When deciding on a product purchase interface for customers, they will see many prototypes first so as to choose the most appealing one. The user feedback is taken for the prototype released and then the work product is produced.

e) Spiral Model:-
Ghantt chart Software, Evolution of Microsoft Windows operating System

GanttPRO software team applied some principles of Spiral Models. For ex:- shorter iteration. to make more frequent releases in order to receive feedback more quickly. Besides a detailed plan describing what to develop for just one iteration was created. Other requirements were documented in the backlog or roadmap

f) Concurrent Models:-
Banking Systems, Travel reservation System, Railway Networks.
Multiprograming, or running a lot of programs concurrently.

g) Evolutionary Model
Ghantt Chatt Software, Evolution of Microsoft windows operating systems

Ganth Bio software team applied some principles of Spiral Model. For ex:- Shorter iteration to make more frequent releases in order to receive feedback more quickly. Besides an detailed plan describing what to develop for just one iteration was created

h) Specialised Process Model

1. Component Based Development:-
Enterprise Java Beans (EJB)
Component Object Model (COM)
.NET model.
CORBA

2. Formal Methods Model.
Medical.
i) Cardiac Pacemaker
Automated Railway System
Unnamed Aircraft System.
Cyber Security etc

3. Aspect Oriented Software development
IBM WebSphere Application Server
JBoss Application Server
Sun Microsystems

(i) The Unified Process :-
This process is included in IBM Rational Method Composer (RMC) product.

Assignment 3

1. Differentiate between Agile Software Model and traditional Software Development

	Traditional Software Development	Agile Software Development
1.	It is used to develop the simple software.	It is used to develop complicated software.
2.	In this methodology, testing is done once the development phase is totally completed.	In this methodology testing and development processes are performed concurrently
3.	It provides less security.	It provides high security.
4.	It provides less functionality in the software.	It provides all the functionality needed by users.
5.	It is basically used for by freshers.	It is used by professionals.
6.	Development cost is less using this methodology.	Development cost is high.
7.	It majorly consists of 5 phases.	It majorly consists of 3 phases.
8.	It is less used by software development firms.	It is normally used by software development firms.

2. Explain with example how principles of agility gives an edge in business.

Business Agility gives the principle of agile development to the entire organisation. This allows companies to be more responsive to change, hasten the time to market and reduce costs without sacrificing quality

Example:-

1) An example of company using business Intelligence.
With business intelligence, a company can receive actionable data on an array of process,

2) Having an e-commerce site can have business agility. With each feedback from customer, the site can be improved to meet the demands.

And also each new iteration will provide new features also.

In all scenarios of the enterprise, where process progress in a waterfall approach, from one phase to another and are getting tied up in a bureaucracy where there are hand-offs from one department to another, org. are instead looking into breakdowns those barriers between dept. by setting up cross functional teams to collaborate on projects and to deliver them rapidly.

3. Different b/w extreme programming and endless-trial XP programming.

	Scrum	XP
1.	In scrum framework team work in iterations called sprint which are 1-2 month long. Scrum model do not allow changes in their timeline or their guidelines.	In XP teamwork for 1-2 week only
2.	Model do not allow changes in their timeline or their guidelines	It allows changes in their set time-lines
3.	It emphasizes self organization.	It emphasizes strong engineering practices
4.	The team determines the sequence in which the product will be developed.	The team have to follow strict priority model or pre determined order.
5.	It is not fully described.	It can be directly applied to team.

4. Discuss XP values.

Ans:- XP is based on 5 values:-

1) Communication:- Communication plays a major role in the sucess of a project. Problems with projects often arise due to lack of communication. Extreme programming does not depend on extensive documentation. As a matter of fact it is suggested only when necessary. So the methodology highly depends on communication between team members and also with users.

2. Simplicity:- This is at the core of XP. The methodology favours simple designs, not thinking too far ahead into the future, but focusing on the requirements of today. while making the program itself robust enough to add the requirements of the future throws up.

3. Feedback:- This essential loop of going back and forth differentiates Agile systems in general and Extreme programming in particular, from other software project management methodologies. The continuous feedback can work in a different ways, but they all work towards making the system stronger and more reliable.

Feedback can come in different flavours.

- From the program itself.
- From the client.
- From the team.

4. Courage:- This might seem a little strange value. Software projects have long been bogged down by traditional XP methods of management; sever

comfort of extreme documentation and hierarchy that doesn't allow for innovation. This values exemplifies the core of XP. Be ready to jump without a parachute if it comes to that.

5. Respect:- Respect the 5th value, was added later and means respect for the code being written and for the clients expectations and needs. This values underlies the communication between different stakeholders as well.

5. Discuss XP process

Ans:- Basic activities that are followed during software development by using XP model are:-

- Coding :- The concept of Coding which is used in XP model is slightly different from traditional of coding. Here coding activities includes drawing diagrams (modeling) that will be transformed into code, scripting web based system and choosing among several alternative solutions.

- Testing :- XP model gives high importance on testing and considers it be the primary factor to develop a fault free software.

- Listening :- The developers need to carefully listen to the customers if they have to develop a good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So it is desirable for the programmers to understand properly the functionality of the system and they have to listen

to the customers.

- Designing :- Without a proper design, a system implementation becomes too complex and very difficult to understand the solution thus makes maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.

- Feedback :- One of the important aspects of XP model is gain feedback to understand the exact customer needs. Frequent contact with customer makes development effective.

- Simplicity :- The main principle of XP is to ~~model~~ develop a simple system that will work ~~together~~ efficiently in present time, rather than trying to build something that would take time and it may never be used - It focuses on simple specific features rather than engaging time and effort on speculations of future requirements.

6. Explain with example following process models :

a) Adaptive Software Development :-

ASD is a method to build complex software and systems. ASD focuses on human collaboration and self organization. ASD "life cycle" incorporates 3 phases only :-

1. Speculation :-

During this phase project is initiated and planning is conducted. The project plan uses project ~~information~~ initiation information like

project requirements, user needs, customer mission statements etc to define set of release cycles that the projects wants.

2. Collaboration:- It is difficult part of ASD as it needs the workers to be motivated. It collaborates communication and teamwork but emphasizes individualism as individual creative plays a major role in creative thinking. People working together must trust each other to:-
   - criticize without animosity.
   - Assist without resentment
   - work as hard as possible.

3. Learning:-
   The workers may have to overestimate of their own understanding of the technology which may not lead to the desired result learning helps the workers to increase their level of understanding over the project.

   Example:-
   The Agile methods mostly follow this approach, with each one having its own rules.

b) Scrum:-
- Scrum is a process framework used to manage product development and other knowledge work.
- Scrum is empirical in that it provides a means for teams to establish hypothesis of how they think something works, try it out, reflect on the experience and make appropriate adjustments
- Scrum is structured in a way that allows

teams to incorporate practices from other frameworks where they make sense in team context.

- Each scrum consists of a number of iterations of work called sprint.
- Example of Scrum:-

Before starting the first sprint one person named "Alex" is assigned as the scrum Product Owner of a new software development process. One of the first task is to start requirement engineering. He writes down the most important use-cases and discusses them with the architects. After collecting the high-level use cases and requirements, he writes them into the Scrum product backlog and initiates an estimation and prioritization session with the architects and some senior developers. During each sprint a set of requirements from backlog is taken & fixed to accomplish.

c) Dynamic System Development Method :- (DSDM)

- The DSDM is an associate degree agile code development approach that provides a framework for building and maintaining systems.
- DSDM is an iterative code method within which every iteration follows the 80% rule that simply enough work is needed for every increment to facilitate movement to the following increment. The remaining detail is often completed later once a lot of business necessities are noted or changes are required and accommodated

DSDM life cycle:-

1. Feasibility Study:-
It establishes the essential buisness necessitees and constraints related to the applyeng to be designed then asseses whether or not the application could be viable candidate for DSDM.

2. Buisness Study:-
It establishes the use and knowledge necessitees that may permit the applyeng to supply business value; additionally, it is essentiding application design and identifies the maintainability necessitees for the applyeng.

3. Functional Model Iteration:-
It produces a collection of progressive prototypes that demonstrate practically for the client.

4. Design and Build Iteration:-
It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner that may altee it to supply operational buisness price for finish users.

5. Implementation:-
It places the newest code increment into the operational surroundings. It ought to be noted that
a. the increment might not be 100% complete
b. changes are also requested because the increment is placed into place. In either case, DSDM development work continues by returning to the

model iteration activity.

d) Crystal :-

- Crystal is an agile methodology for software development. It places focus on people over process to empower team to find their own solutions for each project rather than being constricted with rigid methodologies.

For example :- A small team can keep itself aligned with a regular communication, so it does not need much status reporting and documentation, whereas a large team is likely to get out-of-sync and would benefit from a more structured approach.

Key principles :-

1. Frequent delivery
2. Reflective - Improvement
3. Osmotic communication
4. Personal Safety
5. Focus on work
6. Access to Subject Matter Experts & users.
7. Technical tooling.

e) Feature Driven development :-

- It is an agile development framework that, as its name suggests, organizes software development around making progress on features.
- Features in the FDD context, though, are not necessarily product features in the commonly understood sense, they are rather, more akin to user stories in Scrum.

In other words, "complete the login process" might be considered a feature in the feature driven development methodology.

- FDD Strengths

1) Simple five step process allow for more rapid development.
2) Allows larger teams to move product forward with continuous success.
3) Leverages pre-defined development standards so teams are able to move quickly.

- FDD weakness:-

1) Does not work efficiently for smaller projects
2) Less written documents, which can lead to confusion.
3) Highly dependent on lead developers or programmers.

f) Lean Software Development:-

- Lean Software Development is an agile framework based on optimizing development time and resources eliminating waste, and ultimately delivering only what the product needs.
- The lean approach is also often referred to as the Minimum Viable Product (MVP) strategy in which a team releases a bare minimum version of its product to the market and learns from users what they like, don't like and want to be added, and then iterates based on this feedback.

- LSD strengths :-
1) Streamlined approach follows more functionality to be delivered in less time.
2) Eliminates unnecessary activity, and as a result can reduce costs
3) Empowers the development team to make decisions which can also boost morale.

- LSD weakness :-
1) Heavily depends on the team involved, making it not as scalable as other frameworks.
2) Depends on strong documentation, and failure to do so can result in development mistakes.

g) Agile Modeling :-
- Agile Modeling is one of the agile project development methodologies based on principles of model driven development. Using Agile Modeling techniques and tools allows software developers to consider complex problems before addressing them in programming.

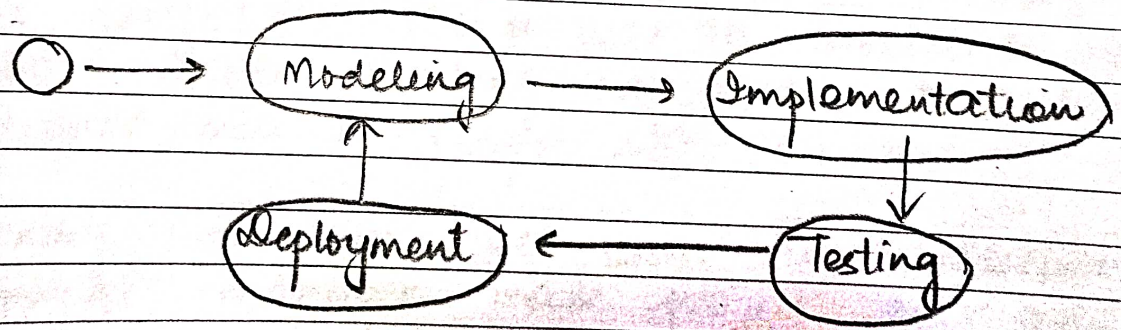
Values of AM
1. Communication
2. Simplicity
3. Rapid Feedback
4. Humility.

It is based on ideas.
1. Content is more important than representation
2. Embracing incremental ~~needed~~ changes in the system.

3. Following agile requirements management
4. Designing multiple effective software models
5. Focusing on quality of agile software lifecycle management and other.

Examples:-
The most popular examples are:-
1) Scrum
2) extreme Programming (XP)
3) Feature Driven Development (FDD)
4) Dynamic System Development Method (DSDM)

h) Agile Unified Process:-
- Agile Unified process is an agile version of Rational unified process.
- AUP is an iterative process consisting of four subprocesses or workflows.

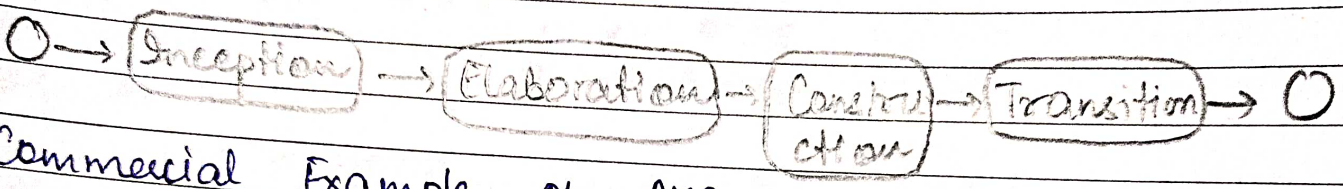


- The last 3 workflows are self explanatory and also appear in RUP.
- AUP combines Business Modeling, Requirements and Analysis & Design into a Agile modeling work flow. Agile modeling minimalist throw away

Models.

- In addition AUP defines the same 3 supporting workflows as RUP
  1) Project Management
  2) Configuration Management
  3) Environment.

- As in RUP, AUP workflow goes through four phases:-



Commercial Example of AUP:-

- A large bank in Greece adopted AUP as its development methodology to implement an important Service oriented Architecture (SOA) project.
- The project objective was to provide a way to host private banking application that can be accessed via a single sign-on.
- During inception use cases were discovered & project manager made project cost and schedule estimates.
- During elaboration, the analyst gathered information while the development team created a mock up showing the UI as screenshots.
- During transition, the team focussed on moving the system into production.

Assignment - 4

1. Explain different levels of design model.

Data / class Design:- It transforms the class models into design class realization and the requisite data structure required to implement the software. The object and the relationship defined in CRC diagram and the detailed data content depicted by class attributes and other notations provide the basis for the data design action.

Architectural Design:- It defines the relationship between major structural elements of software, the architectural styles and design patterns that can be used to achieve the requirements defined for the system and the contraints that effect the way in which architecture can be implemented.

Interface Design:- It describes how the software communicates with system that interoperate with it and with humans who use it. An interface implies the flow of information and a specific type of behaviour. Therefore, usage scenarios and behavioral model provide much of the information required for interface design.

Component level design:- It transforms structural elements of the software architecture into a procedural description of software components. Information obtained from the class based models, flow models and behavioral models

2. List different software quality guidelines.

1. A design should exhibit an architecture that
   1) has been created using recognizable architectural styles or patterns
   2) is composed of components that exhibit good design characteristics.
   3) can be implemented in an evolutionary fashion.

2. A design should be modular, that is, the software should be logically partioned into elements or subsystems.

3. A design should contain distinct representations of data, architecture, interface and components.

4. A design should lead to data structure that are appropriate for the classes to be implemented and are drawn from recognizable data patterns.

5. A design should ~~be~~ lead to components that exhibit independent functional characteristics

6. A design should lead to interfaces that reduce the complexity of connections b/w components and external environment.

7. A design should be derived using a repeatable method that is driven by information obtained during software requirement analysis.

8. The design should be represented using a notation that effectively communicates meaning.

3. Explain quality attributes given by Hewlett Packard.

Hewlett Packard developed a set of quality attributes that has been given the acronym FURPS – functionality, usability, Reliability, performance and Supportability.

- Functionality:- It is assessed by evaluating the features set and capabilities of the program the generality of the functions that are delivered and the security of the overall system.
- Usability:- It is assessed by considering human factors, overall aesthetics, consistency and documentations
- Reliability:- It is evaluated by measuring the frequency and severity of failure, the accuracy of result, the mean time-to-failure (MTTF), the ability to recover from failure and the predictability of the program.
- Performance:- It is measured by considering processing speed, response time, resource consumption throughput and efficiency.

Supportability:- It combines the ability to extend the program, adaptability, serviceability - these 3 attributes represent a more common

term, maintainability - and in addition, testability, compatability, configurability, the ease with which a system can be installed, and the ease with which problems can be localized.

Q4 Explain design concepts by mentioning following
1) definition
2) Importance
3) Example

a) Abstraction:-
When you consider a modular solution to any problem, many level of abstractions can be posed. A procedural abstraction refers to a sequence of instructions that have a specific and limited function.
Data abstraction is a named collection of data that describes a data object.
Importance:- It helps the user to only get the required information without going into much details.
Example:- While booking a particular items from an e-commerce site the user should not be aware of the inner detail.

b) Architecture:-
One goal of software design is to derive an architectural rendering of a system. This rendering serves as a framework from which more detailed design activities are conducted.

Importance:- The derivation of architecture is an integral part of the design process. The manner in which the software architecture is characterized and its role in design are very important.

Example:- A number of ADL's have been developed to represent different models. The way different modules have been designed is an example of architecture.

c) Pattern:- A pattern is a named nugget of insight which conveys the essence of a proven solution to a recurring problem within a certain context amidst competing concerns.

Importance:- In case of any problem, the predefined patterns can be easily used to solve

Example:- In case of requirement gathering communicating with stakeholders can be useful.

d) Separation of Concerns:- Separation of concerns is a design concept that suggests that any complex problem can be more easily handled if it is subdivided into pieces that each can be solved and or optimized independently.

Importance:- Separation of concerns can help in isolating and finding errors easily.

Example:- problems P1 & P2 if the perceived complexity of P1 is greater than the perceived complexity P2, it follows that the effort required to solve P1 is greater than to solve P2.

e) Modularity:- Software is divided into separately name and addressable components, sometimes called modules, that are integrated to satisfy problem requirement.
Importance:- It allows the program to be intellectually managable.
Example:- In an e-commerce website they will be different modules for payment and for selection of product being added to cart.

f) Information hiding:- This is the concept which says that the non required details should be hidden from user because it increases the level of complexity.
Importance:- Hiding implies that effective modularity can be achieved.
Example:- In case any error occurs in payment module it can be solved easily because it won't effect other.

g) Functional Independence:- It is achieved by developing modules with single minded function and an "aversion" to excessive interaction with other modules. It involves two concepts cohesion and coupling.
Importance:- Simple connectivity among softwares modules results in software that is easier to understand and less prone to ripple effect.
Example:- In Safe Home Architecture only necessary linking of functions should be done.

h)

h) Refinement :- A program is refined by successf refining concepts / levels of procedural details
Importance :- It enables us to specify procedura and data internally but suppress the need o outsiders to have knowledge of low level details.

(i) Refactoring :- It is a reorganization technique that simplies the design of a component without actually changing its function or behaviour.
Importance :- It allows us to check for redu dancy, unused elements, inefficient or une- necessary algorithms, poorly constructed or inappropriate data structures.
Example :- A first design iteration might yield a component that exhibits low cohesion. Af ter careful consideration, you may decide the components should be refactored.