**OOAD**

**Analysis** -> What is the big picture of the application that we want to develop

It focuses on the requirements of the customers

It focuses on the architecture , technology,language

It focuses on non-functional requirements

(portability,scalability , adaptable , extensibility ,

performance , reliability)

**Design** -> How to develop the application to meet the requirements of

the customers such that both functional and non-functional

requirements will be taken care of.

**Challenges of the existing project**

- Software Complexity

- Software Decomposition

- Software Cost(Cost towards development/resources,cost towards the lincensed softwares/tools, Cost towards maintainence and support,cost towards organizing the training programs to enhance the skillset of the developers)

**Project Life Cycle?**

- Requirements capturing -> Customer/Business Analyst

- Analysis -> Architect

- Design -> Designer

- Development/Coding -> Developer / Tech Lead

- Testing -> Testing team/Quality Assurance team

- Deployment -> Deployer/Integrator

- Maintainence/Support/Enhancement -> Support Engineer

**Reasons for Software Complexity**

- User need is not met

- Requirements change

- Modules don't integrate

- Code hard to maintain

- Late discovery of the flaws/errors

- Poor quality and performance check

- Build and release problems

**Root Causes for Software Complexity**

- Insufficient requirements

- Communication gaps between the stakeholders

- Brittle Architecture

- Complexity in the domain

- Poor testing

- Waterfall model

**Best Practices to follow in a Software Development Cycle**

- Develop software iteratively and incrementally

- manage requirements

- Visually model the requirements/software to be developed (UML)

- Build component driven architecture

- Continously verify the quality of the software

- Control changes

**OOSD - Object Oriented Software Development**

**The entire software is divided into phases**

Phase 1 - Inception (Requirements project life cycle stage)

Phase 2 - Elaboration ( Analysis and Design project life cycle stage)

Phase 3 - Construction (Development project life cycle stage)

Phase 4 - Transition (Testing,Deployment,Maintainance project life cycle stage)

**Phase -> workflows -> activities -> model -> UML diagram**

**Requirements** -> Requirements Model -> UseCase diagram

**Analysis** -> Analysis Model -> Class diagram/Object diagram(High Level Design)

**Design** -> Design Model ->Class Diagram/Object Diagram(Low Level Design)

**Development** -> Development Model -> Interaction Diagram

a) Sequence diagram

b) Collaboration diagram

-> StateChart diagram

-> Activity diagram

-> Component diagram

**Testing** -> Test Model -> Test Cases

**Deployment** -> Deployment Model -> Deployment diagram

**Model?**

- It is a representation of the reality

- It is conceptual

- Helps to understand ,visualize , construct , document the artifacts of

the system that we are developing

**UML - Unified Modeling Language**

Classified into 2 types

**a) Static Diagrams**

- usecase diagram

- class diagram

- component diagram

- deployment diagram

**b) Dynamic Diagrams**

- Interaction Diagram

a) Sequence diagram

b) Collaboration diagram

- StateChart diagram

- Activity diagram

**History of UML**

- In 1970 -> Ericcson -> blocks -> block diagram

- in 1980 -> Ivar Jacobson -> renamed block diagram as component diagram

- In 1987 -> Ivar Jacobson started a organization called Objectory AB

-> focus on developing model , workflow and activities

- In 1995 -> Rational Software Corporation merged with Objectory AB

-> Grady Booch -> phases

-> 4+1 Views of the Software Architecture

-> Brought in a concept of standardizing the symbols and concepts

of OOSD

-> This gave the birth to a process called Rational Unified Process(RUP)

-> first verion of UML 1.0 started

**4+1 View of the Software Architecture**

a) Logical View -> classes , interfaces , relationships between the class

b) process view -> threads to be created and the synchronization and concurrency features

c) Development view -> component and their relationships

d) Deployment view -> systems architecture , environment and deployment mechanism

e) usecase view -> requirements -> usecase model/usecase diagram

**Stakeholders in 4+1 View of the Software Architecture**

a) UseCase View -> Customer / Business Aanalysts

b) Logical View -> Arcchitecture / Designer / Tech Lead

c) Process View -> Architect/Designer

d) Development view -> Developer

e) Deployment view -> Testing Team/Quality Assurance Team /Integrator / Deployer

**Relationships between the usecases**

**a) include** -> it is a relationship between 2 usecases and one usecase is dependent on another usecase to perform its task

**b) extends** -> it is a relationship between 2 usecases(example validating pin and performing a transaction) Here it is not mandatory for the usecase called perform transaction to be performed after the pin number is validating

**c) generalization** -> It is a relationship between 2 usecase with a inheritance relationship

**Use Case Specification Document/Supplementary Document**

Every use case should contain

1. Name
2. Brief Description
3. UseCase Diagram to show its relationship with other use cases and actors(if existing)
4. Pre condition
5. Post condition
6. Flow of events
   1. Basic/Success Flow
   2. Alternate/Failure Flow
7. Exception Flow
8. Special Requirements
9. Extension points