SOFTWARE ENGINEERING

DITU (2021-22) B.Tech 3rd year

Software Engineering

- Systematic approach to design, development, operation & maintenance of a software system.

Objectives :-

- Maintenance
- Correctness
- Reliability
- Testability
- Reusability
- Adaptability
- Portability

(MCRTRAP)

Dual role of software

- As a product
  - Provide computing potential, enables H/w to produce desired functionality, info transformer
- As a vehicle for product delivery
  - system functionality (payroll system), controls other s/w (O.S), build other software (s/w tools)

S/w characteristic

1. functionality (Suitability, Accuracy Interoperability, Compliance, Security)
2. Reliability
3. Efficiency (T, R)

(FREUMP)

4. Usability (Understand, Learn, Operate)
5. Maintainability (Test, Stable, Change, Operate)
6. Portability (Adaptibility, Installability, Replaceability)

Software Crisis

- Increasing demands, complexity, challenge
- same workforce, methods, tools
- S/w not meeting requirement
- ↑ expenses
- S/w not delievered
- Projects running overtime

Components of S/w →

- Program
- Documentation
- Operating Procedures

Software engineering process

- S/w specifications
- S/w development
- S/w validations
- S/w evolution

(SDEV)

SDLC Model

- Classical Waterfall (FSDCIM)
  - * System Testing (Alpha, beta, acceptance)
  - * Maintenance (Corrective, Perfective, adaptive)
- Prototyping Model
- Spiral Model / Meta Model (+Risk Handling)
  - * Radius → Cost, Angular dimension → Progress (Phase)
  - * ORVR (Objective, Risks, Versions, Reviews)
- Iterative Enhancement
- Evolutionary

Requirement Elicitation

- Interview
- Brainstorming Sessions
- fAST
- QfD
- Use case approach (what, not how)

- Use case → Actor, use cases, diagram (Primary, Secondary)

🧍 → Actor , ⬭ → Use case , — → relation.

Requirement Analysis & Documentation

- Context Diagram
- Prototype (optional)
- Model requirements
- finalise requirements

Why software Engineering?

- large software
- Scalability
- Cost
- Dynamic Nature
- Quality Management

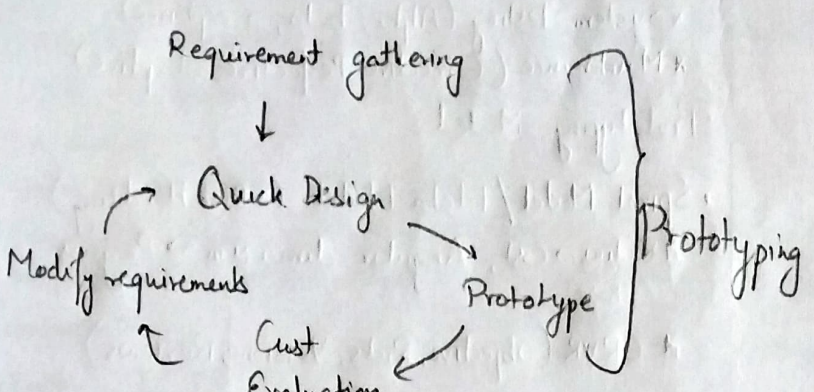
Solutions (Software Engineering)

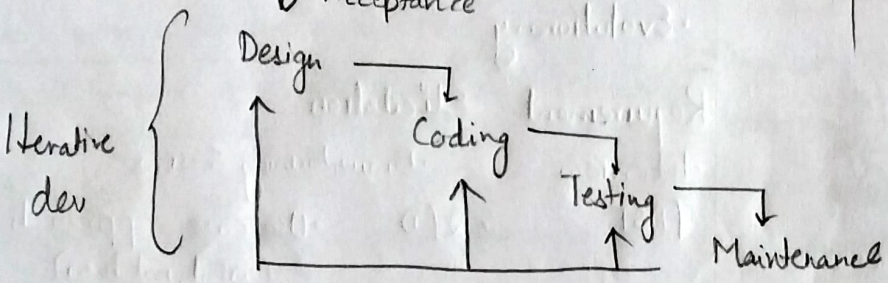
- → Reduction in over budget
- → High quality product
- → Deadlines
- → Experienced member on team
- → Timely delivery

Software Process Models

1. Workflow Model
2. Waterflow Model (S → D → I → T)
3. Dataflow Model
4. Evolutionary Development Model (S → D → V)
5. Role/Action Model

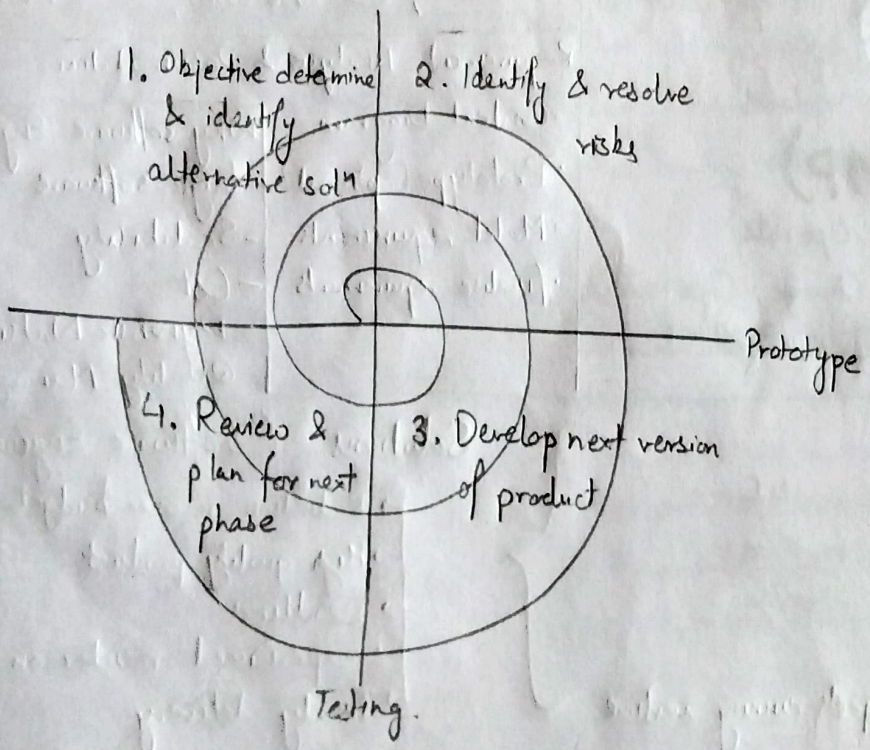
Prototype





- fAST
  - facilitated application specification technique
- QfD
  - Quality funct^{n} deployment
- Interview
  - Open ended, structured.
- QfD
  - Normal requirement
  - Expected requirement
  - Exciting requirement

Spiral



Risk handling
↳ Build prototype in each phase

① Entity-Relationship Diagram

- Data modeling method
- Produce a conceptual data of an info system.
- Better understanding of data to be contained in the database
- Serves as a documentation tool
- Connects the logical structure of database to users.
- Visual representation of different entities within system & how they're related

Components

① Entity ▭

- A real world object which exists, is distinguishable & has a state & behavior at all times

Entity set: Collection of related types of entities (need not to be disjoint)

② Attributes

- Properties of entity having a value (domain/range assigned)

Types

i). Key - 1 or more attributes uniquely identifying an entity
- → Super key (set of attributes collectively identifying entity)
- → Candidate key (minimal super key)
- → Primary key (the candidate key chosen to identify entity set)

ii). Composite - Composition of other attributes

iii). Single-valued - Contains a single value

iv). Multi-valued - Have more than 1 value ⬭

v). Derived - Doesn't exist in DB, but can be derived from other present attributes. (- - -)

③ Relationships ◇

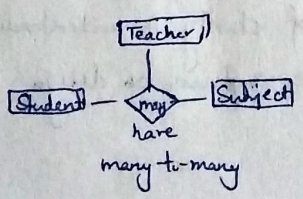
- Association among entities
- Attributes of relationships → Descriptive attributes.

* Degree of relationship set
- No. of participating entities in a relationship.

i). Unary → recursive (1 person is married to 1 person only)

ii). Binary (Teacher teaches subject)

iii). Ternary

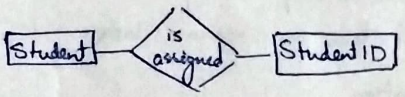


④ Cardinality

- No. of entities in one entity set, which can be associated with no. of entities of other sets via relationship set.

Types

i) One-to-One



ii). One-to-Many

iii). Many-to-One

iv) Many-to-Many

Advantages

- Conceptually simple
- Better visual representation
- Effective communication tool
- Highly integrated with relational model
- Easy conversion to any data model

Disadvantages

- Limited constraints & specifications
- Loss of info content
- Limited relationship representation
- No industry standard for notation

② Use-case diagram

- Summarize details of your system's users (actors) & their interactions with the system.
- Makes use of specialized symbols & connectors
- Represent situations where your system interact with people, organizations, etc. & help them achieve goals.
- Depicts a high-level overview of relationship b/w use cases, actors & systems.

Used in
- Representing goals of system-user interactions
- Defining & organizing functional requirements in system
- Specifying context & requirements of system
- Modeling basic flow of events

Components

① **Actors** 

- Users interacting with system
- Can be a person, organization or an external system
- Produce or consume data

② **System**

- Specific sequence of actions & interactions b/w actors & system

③ **Goals**

- End result of use case
- Describe activities & variants to reach the goal

④ **Use-case** 

- Different uses an actor might have

⑤ **Associations**

- line b/w actors & use cases

⑥ **System boundary boxes**

- Sets a system scope for use cases

Advantages

- Easily understandable (composed of narrative text)
- Early feedback from end users
- Identification of exceptional scenarios for use cases
  ↳ discover subtle alternate requirements in the system
- Comprehensive summary in 1 single illustration

Disadvantages

- Might not capture non-functional requirements easily.
- Not object oriented
- Lack of formality
- Issues when a functionality doesn't require an actor

③ Class Diagram / Structural diagram

- Represent static view of applications
- Describe attributes & operations of a class & also the constraints imposed on the system
- Mostly used in modeling of object oriented systems
- Shows a collection of classes, interfaces, associations, collaborations & constraints

Points to be remembered

- Name of class diagram should be meaningful to describe aspect of the system.
- Each element & their relationships be identified in advance
- for each class, min no. of properties be specified
- * Order & Customer identified as 2 elements of the system. They have 1-to-many relationship
- * Order class → abstract class → inheritance relationship

Components

- Upper section → Class name
- Middle section → Attributes of classes (describe instances)
- Bottom section → Class operations (methods)
- Member access modifiers: Public (+) Private (-) Protected (#) Package (-) Derived (/) Static (underline)
- Member scopes
  - Classifiers (static members)
  - Instances (specific instances)
- Signals (one-way, asynchronous communications)
- Interactions
  links exist in class & object diagrams.

Inheritance → ↑

Bidirectional - ——

Unidirectional - →

Advantages

- Illustration to acquire max. info
- Schematics of application understandable
- Any requirement can be highlighted & programmed to described structure
- faster, easy to ready & create

Disadvantages

- Longer time for management & synchronization
- Lack of clarity in understanding beneficiary
- Puts over-emphasis on design

④ Sequence Diagram / Event diagrams

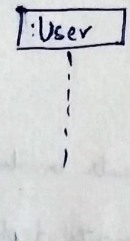
- Interaction diagram, describing how & in what order a group of objects work togethers
- Used to understand requirements for a new system or to document an existing process.

Components

i)  → Object (represent class/object not attributes)

ii) → Activation box (time needed to complete task)

iii)  → Actor (entities interacting with system)

iv) 

→ Lifeline (represent passage of time, as it extends downwards; dashed line shows the sequential events that occur to an object during charted process).

v) Synchronous msg symbol (sender must wait for a response before continuing)

vi)  Asynchronous msg symbol

vii) 

Msg. creates a new object

Advantages

- Explore any real application/system
- Represent msg. flow from one object to another
- Easier to represent & generate
- Easily update acc. to changes within system
- Allows reverse & forward engineering

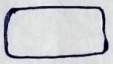
Disadvantages

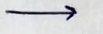
- Can become too complex
- Order of msg. should be conserved.
- Each sequence represent different msg. notation.
- Type of msg. decides type of sequence inside diagram.

⑤ State Chart Diagram

- Stores status of an object at a given time & accomodate changes based on input.
- States refer to different combinations of info that an object can hold, not how the object behave
- Depict states & transitions

Components

 → State

 → Transition

 → Initial state

 → final state

Advantages

- Ideal way to model object lifecycles
- Makes system behavior visible

Disadvantages

- Not applicable for finite systems
- Complexity ↑ as no. of possible states ↑
- Lack of support for concurrent constructs in modelling the system.

⑥ Collaboration Diagram / Communication Diagram

- Illustration of relationships & interactions among software objects.
- Portray dynamic behavior of a particular use case & define role of each object.
- Identify structural elements required to carry out functionality of interaction → model built using relationship b/w those elements.
- Resembles a flow chart that portray the roles, functionality & behavior of individual objects as well as the overall operation of system in real time

Components

① Object → 

② Actors →  (invoke interaction in diagram)

③ Links → connect objects with actors

④ Msg → labelled arrow placed near link

Advantages

- Gives priority to interactions
- Messages conveyed over sequencing
- Object diagram is unique
- Cost-friendly

Disadvantages

- Complex to disclose the objects
- Time exhausting
- Doesn't take into account the state of object

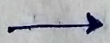
⑦ Activity Diagram

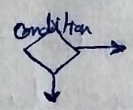
- Behavioral diagram, depicts behavior of system.
- Illustrate flow of control in the system & refer to the steps involved in execution of a use case.
- Depiction of both sequential & concurrent processing of activities using it.

Components

① Initial state 

② Action/Activity state 

③ Control flow 

④ Decision Node 

⑤ End state 

Advantages

- Complex states can be easily explained
- Flows & processes depicted easily
- Each activity flow is explained as it is
- Methods, funct^{n} & operations are explained in detail.

Disadvantages

- Communication b/w 2 components/users can't be shown.

⑧ Deployment Diagram

- Visualize topology of physical components of a system, where software components are deployed.
- Describe static deployment view of a system.
- Consists of nodes & their relationships

Components

① Node 

② DB 

Advantages

- Create better visualization of elements involved
- Better description of all hardware elements deployed for software components
- Describe involved runtime processing nodes for better clarity