Syllabus

22 September 2022 07:07 PM

UNIT-1 **Introduction to Software Development process and Management**
Introduction to Software Engineering, Software Crisis and Myths, Software Development life cycle and Models: Maturity Model, Process models-waterfall, evolutionary, incremental etc, What is an agile view of process.

UNIT-2 **Requirements Engineering**
Requirements Engineering and Management. Initiating, Eliciting requirement, developing use cases, building the Analysis Model, Negotiating and Validating requirement.

UNIT-3 **An Architectural Design**
Software Architecture, Data Design, Architectural Styles and pattern, Architectural Design, Assessing alternative architectural Design Mapping Data flow in to software architecture

UNIT-4 **User Interface Level Design and Estimation**
The Golden rules, User Interface analysis and Design, Interface Analysis, Interface Design Steps, Design Evaluation, Project Estimation.

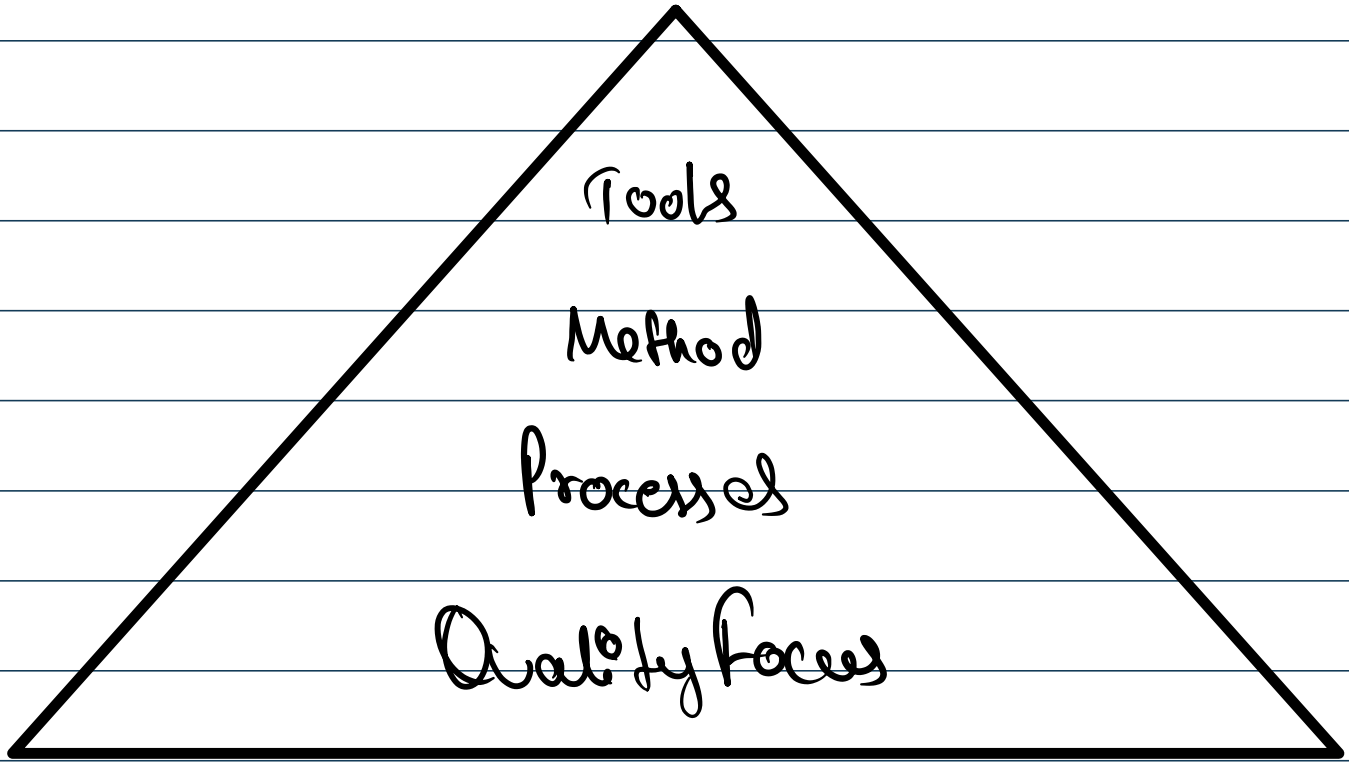
UNIT-5 **Introduction to Software Testing**

8

Test Strategies for conventional software, Validation testing, System Testing, The art of Debugging, Software testing fundamentals, Software quality, Framework for product Metrics.

SE Layered Technology

28 December 2022 08:47 AM



Generic Process Framework

28 December 2022 07:28 AM

Communication

- initiation
- requirement

Planning

- Estimation
- Scheduling
- Tracking

Modeling

- Analysis
- Design

Construction

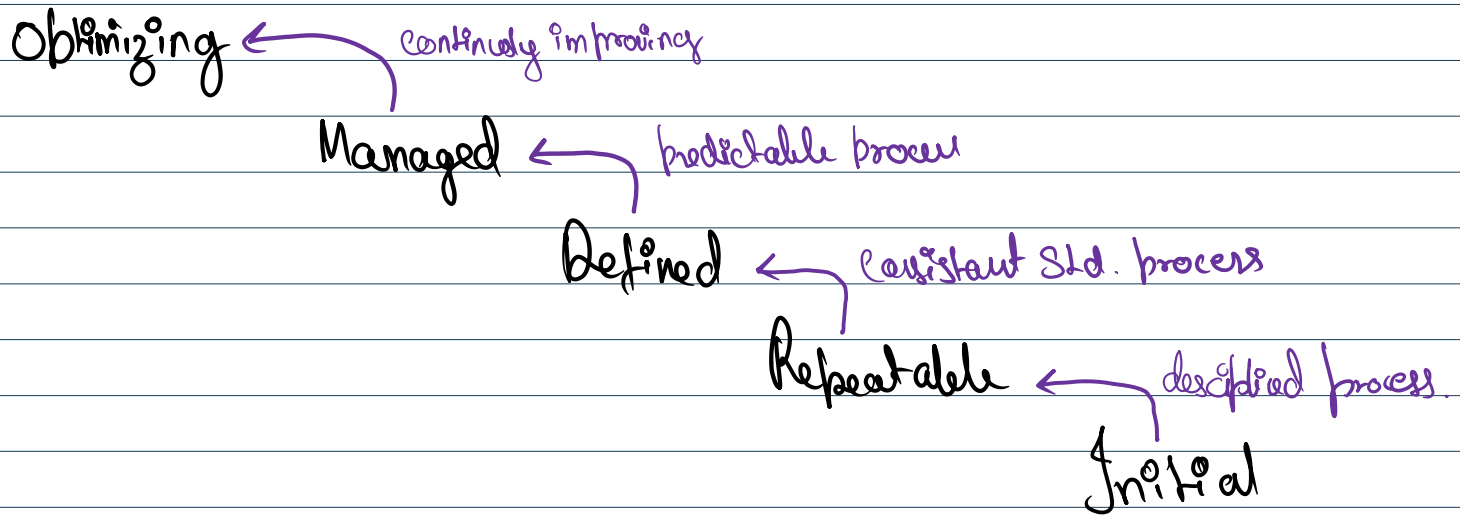
- Code
- testing

Deployment

- delivery
- Feedback

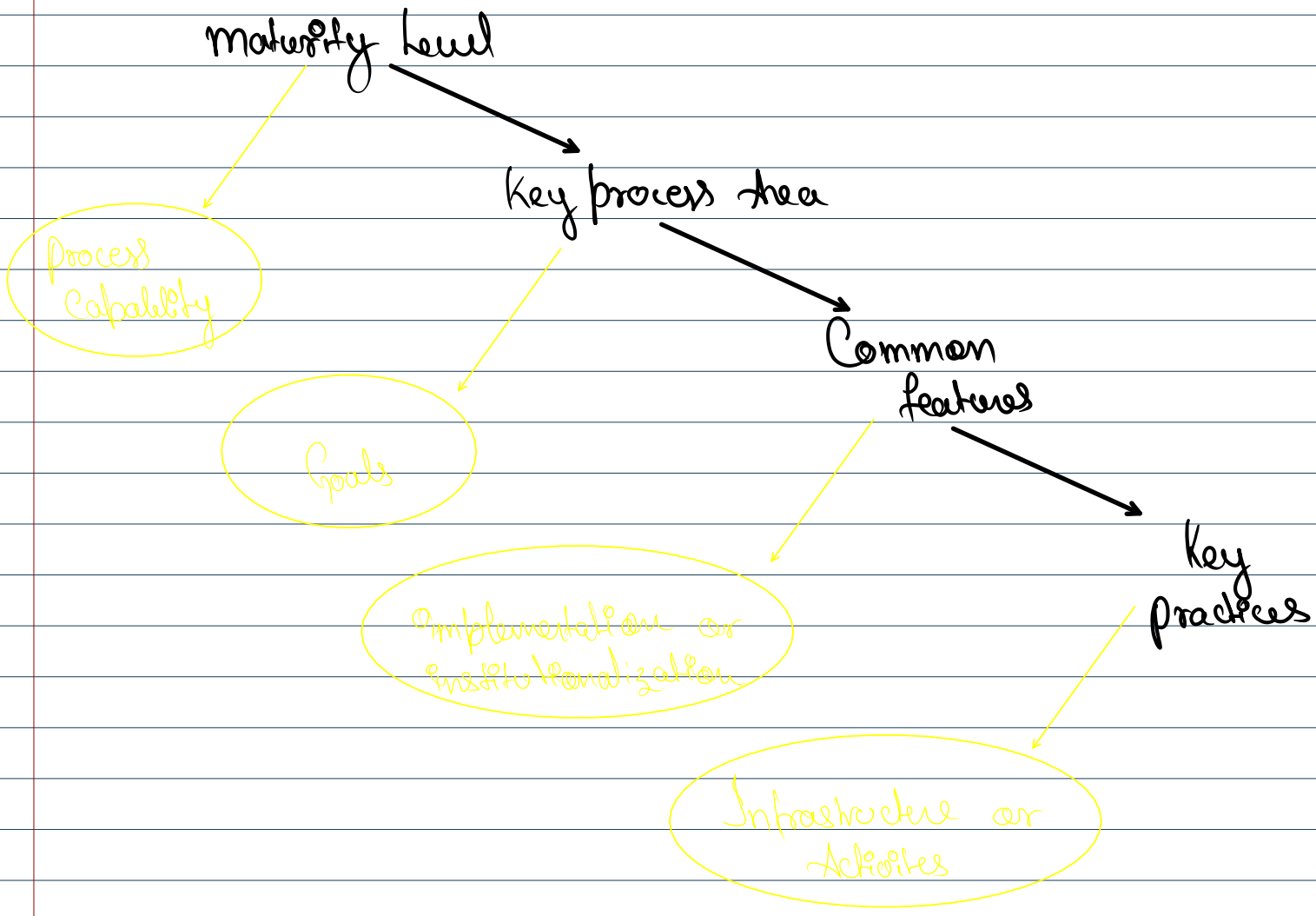
CMM Level

28 December 2022 08:51 AM



CMM Structure

28 December 2022 08:55 AM



Communication Practices (Requirements Elicitation)

28 December 2022 07:28 AM

Communication
- initiation
- requirement

Planning
- Estimation
- Scheduling
- Tracking

Modeling
- Analysis
- Design

Construction
- Code
- testing

Deployment
- delivery
- feedback

Requirement Engineering

28 December 2022 07:56 AM

Inception : Identify Stakeholders, viewpoints, Break Ice, Set goals

Elicitation : Identification of Scope / understanding / validity

- Interview
- Survey
- Questionnaires
- Task analysis
- Domain analysis
- Brainstorming
- Prototyping
- observation

Elaboration : Combine & technicalize from two stages into.

- Usecase
- State diagram

Negotiation ~ Self-Explanatory

Specification ~ SRS

Validation ~ review

Requirement Management

~ tracking

etc. etc.

feed back

Requirement types

1) Normal
2) Expected
3) Exciting

1. Scenario-based modelling ~ user POV (use case)
2. Class-based modelling ~ attri, obj, relation (class diag)
3. Flow-oriented modelling ~ flow ----- (structure)
4. Behavioural modelling ~ depicts states & classes (State diag)

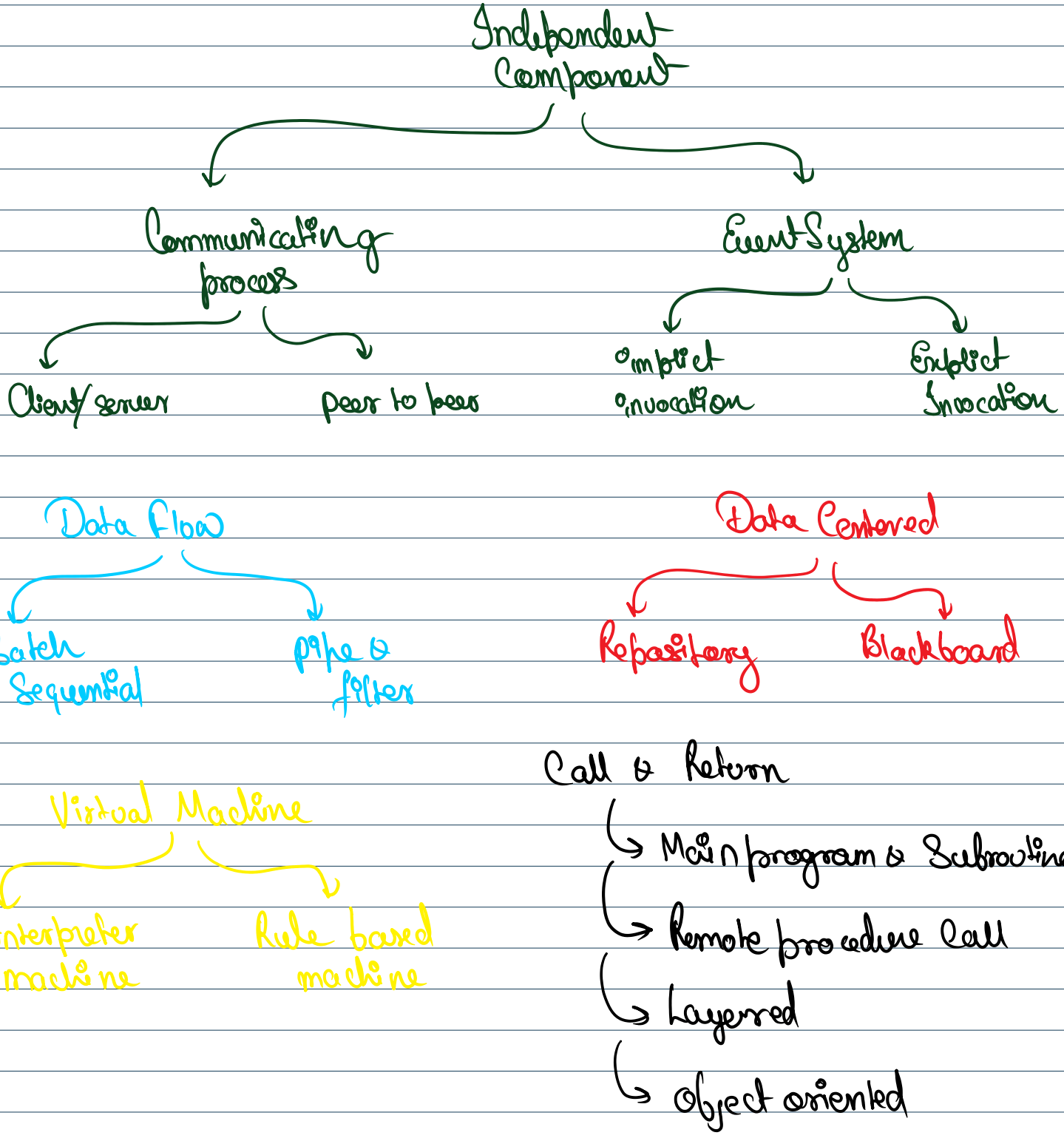
test other or object oriented

Example Class Box

Class Name	Component
Attributes	+ componentID - telephoneNumber - componentStatus - delayTime - masterPassword - numberOfTries
Operations	+ program() + display() + reset() + query() - modify() + call()

Taxonomy of Architectural Styles

28 December 2022 05:44 AM



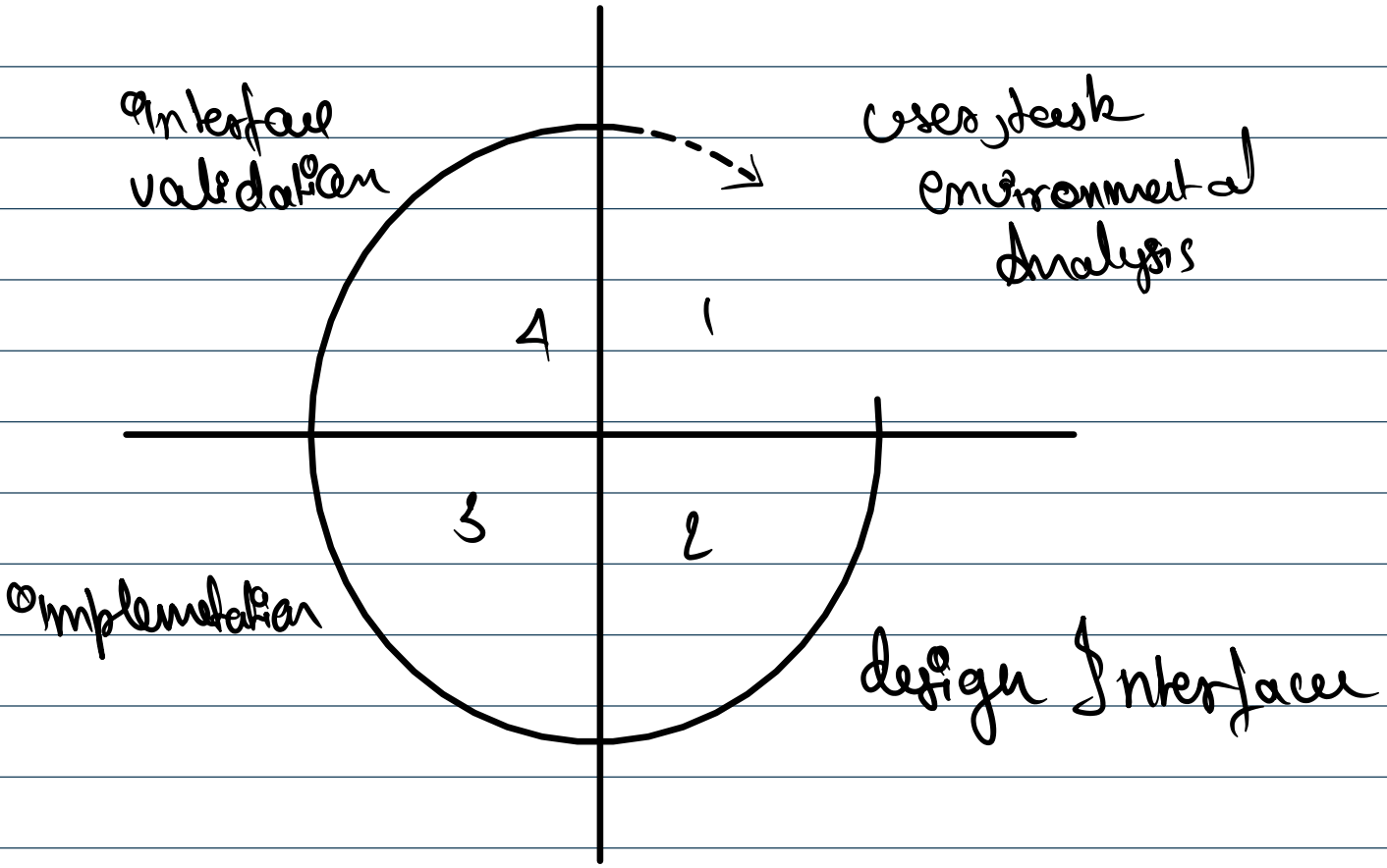
Architectural Design Steps

28 December 2022 07:00 AM

1. Represents the System in Context
2. Define Archetypes
3. Refine Architecture into Components
4. Describe Instantiations of System

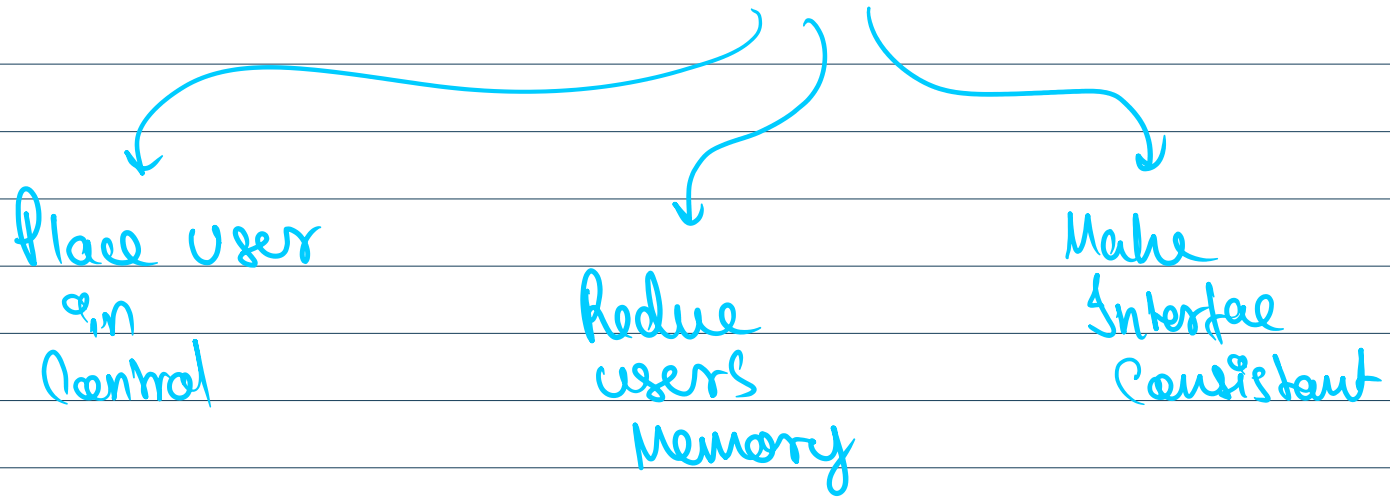
Spiral Diagram for UI Dev

28 December 2022 02:36 AM



28 December 2022 03:00 AM

Golden Rules UI Design



1. User profile Mode (Human Eng. or SW eng.)

2. Design model (SW Eng.)

3. Implementation Model (SW Implementers)

4. User's Mental Model (developed by user yourself)

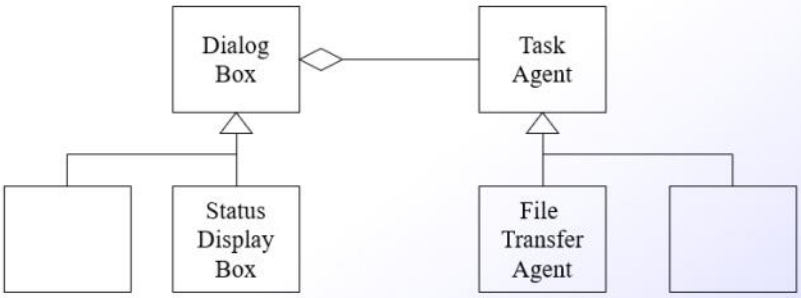
User Profile Model

- Establishes the profile of the end-users of the system
  - Based on age, gender, physical abilities, education, cultural or ethnic background, motivation, goals, and personality
- Considers syntactic knowledge of the user
  - The mechanics of interaction that are required to use the interface effectively
- Considers semantic knowledge of the user
  - The underlying sense of the application; an understanding of the functions that are performed, the meaning of input and output, and the objectives of the system
- Categorizes users as
  - Novices
    - No syntactic knowledge of the system, little semantic knowledge of the application, only general computer usage
  - Knowledgeable, intermittent users
    - Reasonable semantic knowledge of the system, low recall of syntactic information to use the interface
  - Knowledgeable, frequent users
    - Good semantic and syntactic knowledge (i.e., power user), look for shortcuts and abbreviated modes of operation

15

Design Model

- Derived from the analysis model of the requirements
- Incorporates data, architectural, interface, and procedural representations of the software
- Constrained by information in the requirements specification that helps define the user of the system
- Normally is incidental to other parts of the design model
  - But in many cases it is as important as the other parts



16

Implementation Model

- Consists of the look and feel of the interface combined with all supporting information (books, videos, help files) that describe system syntax and semantics
- Strives to agree with the user's mental model; users then feel comfortable with the software and use it effectively
- Serves as a translation of the design model by providing a realization of the information contained in the user profile model and the user's mental model

User's Mental Model

- Often called the user's system perception
- Consists of the image of the system that users carry in their heads
- Accuracy of the description depends upon the user's profile and overall familiarity with the software in the application domain

Verification & Validation

27 December 2022 10:33 PM

Verification

- the algo coded correctly
- if Software correctly implements function / algo.

Validation

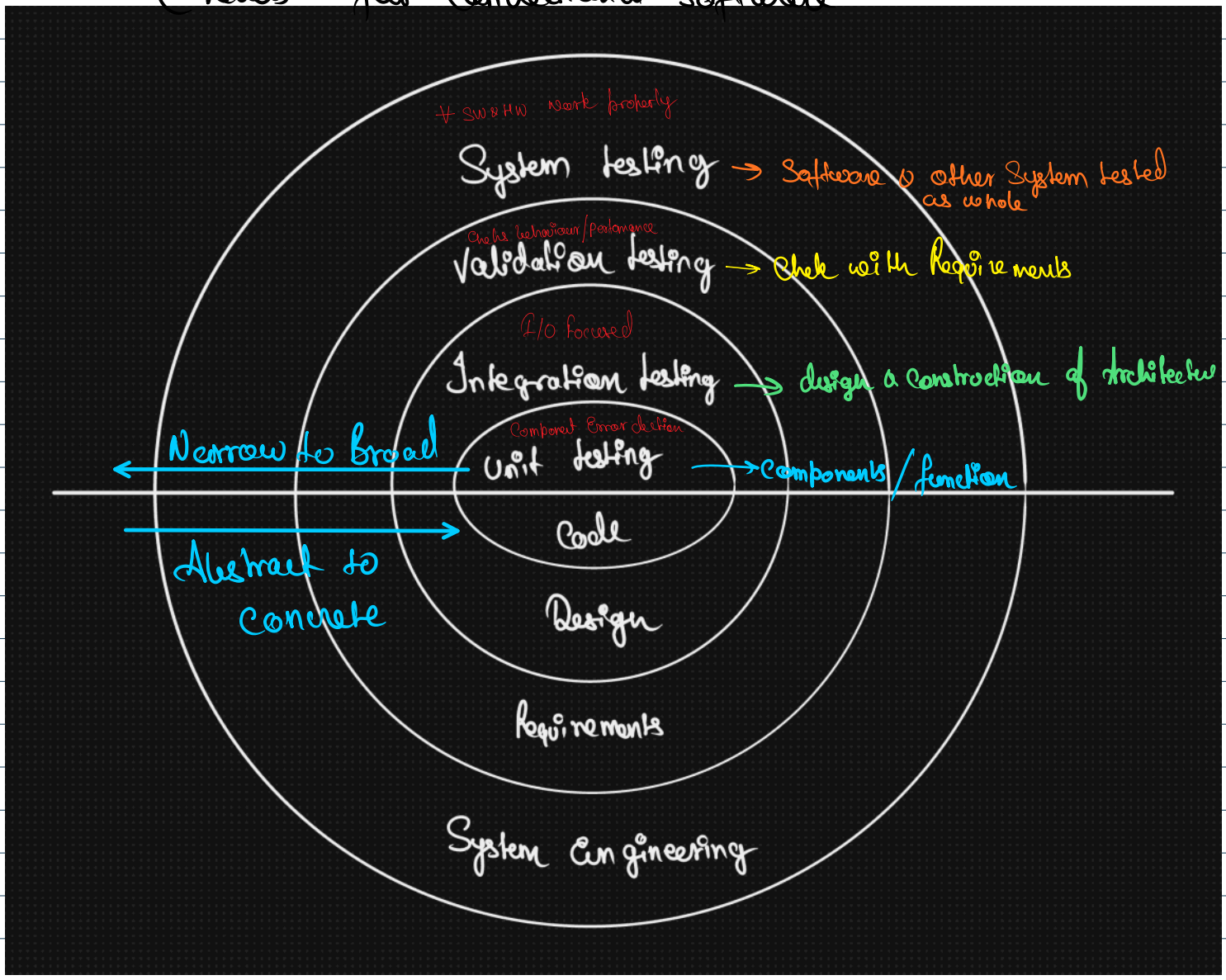
- Does it meet user requirement
- If build software is acceptable to Customer

Includes Software testing

Levels of Convectional SW Testing

27 December 2022 10:37 PM

@ levels for Convectional Software



!Drivers and Stubs

27 December 2022 11:00 PM

Strategy for Conventional SW Testing

27 December 2022 11:10 PM

① Unit testing

↳ @ Components & functions

↳ Internal processing logic & data structures

↳ Concentrate on modules with high cyclomatic complexity

@ targets
- Module interface
- Local data structures
- Boundary conditions
- Independent path (Basis path)
- Error handling path

② Integration testing

↳ @ Constructing SW Architecture

takes unit modules/components & build program structure w.r.t design

Approaches

Non Incremental integration testing

@ Big Bang

Incremental integration testing

@ program constructed & tested incrementally

@ Big Bang

- Chaos
- All Component Combined in advance
- endless loop of error & resolution

top down | Bottom up | Sandwich

- Body ↓ Cell
- Cell ↓ Body
- X

- main module to components
- Establishes controls early on.
- Stubs needed to substitute for lower modules (which aren't built yet)
- no significant data flow due to stubs until much later

- basic components to main module
- No need of stubs
- verifys low level data flow
- Drivers are needed to be built to test low level modules

③ Validation Testing

here distinction between conventional & OO sw disappears

@ focuses on user-visible actions & user recognizable output from the System

↳ Demonstrates Confirmity with Requirements

④ System testing

↳ Recovery testing : Test for reinitialization / Checkpoint Mech / data Recovery

↳ Recovery testing : Test for reinitialization / Checkpoint Mech / data Recovery

↳ Security testing : Verification of protection mechanism

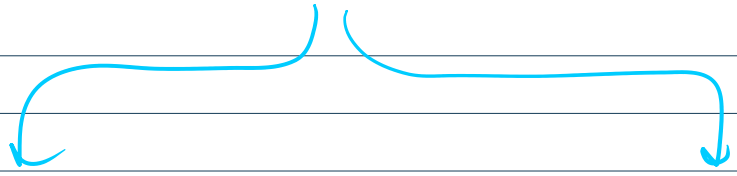
↳ Stress testing : Execution of system in resource demanding manner

↳ Performance testing : @ runtime performance
one HW & SW

Alpha & Beta Testing

28 December 2022 01:57 AM

@ Maybe related to Interface IDK



Alpha testing

@ Developer Site

↳ Controlled Environment

Beta testing

@ End user - Site

↳ User Site Environment @chaotic

Regression Testing & Smoke Testing

27 December 2022 11:27 PM

might be a part of previous topic IDK

① Regression Testing

- Each new addition / change to Software might cause problem with previously working functions
- This test re-executes small test that have previously been conducted (to be on the safe side)
  - ↳ looks for unintended sideeffect of addition

② Smoke Testing

© inspiration → hardware

↳ power is applied briefly to see dramatic sign of fundamental failure

↳ Integration risk is minimized

+ Some Stuff....

- Taken from the world of hardware
  - Power is applied and a technician checks for sparks, smoke, or other dramatic signs of fundamental failure
- Designed as a pacing mechanism for time-critical projects
  - Allows the software team to assess its project on a frequent basis
- Includes the following activities
  - The software is compiled and linked into a build
  - A series of breadth tests is designed to expose errors that will keep the build from properly performing its function
    - The goal is to uncover "show stopper" errors that have the highest likelihood of throwing the software project behind schedule
  - The build is integrated with other builds and the entire product is smoke tested daily
    - Daily testing gives managers and practitioners a realistic assessment of the progress of the integration testing
  - After a smoke test is completed, detailed test scripts are executed

Benefits of Smoke Testing

- Integration risk is minimized
  - Daily testing uncovers incompatibilities and show-stoppers early in the testing process, thereby reducing schedule impact
- The quality of the end-product is improved
  - Smoke testing is likely to uncover both functional errors and architectural and component-level design errors
- Error diagnosis and correction are simplified
  - Smoke testing will probably uncover errors in the newest components that were integrated
- Progress is easier to assess
  - As integration testing progresses, more software has been integrated and more has been demonstrated to work
  - Managers get a good indication that progress is being made

----------X----------

27 December 2022 11:06 PM

- Unlike conventional software in which functions / components have clear defined I/O
- In OO SW doesn't, it has classes which don't have I/O behaviour

due to above reason Unit testing loses meaning

Techniques for OO testing
- ↳ Fault based testing
- ↳ Class based testing
- ↳ Random testing
- ↳ Partition testing
- ↳ Scenario based testing

Testing Strategy for OO SW
- Thread Based
- Use Based

----------X----------

27 December 2022 11:06 PM

?Art of Debugging

28 December 2022 02:07 AM

Debugging is a Consequence of Successful testing

Strategies for debugging

- Brute force @ most common & least efficient
  - → Memory dump, runtime traces, output statements
- Back tracking @ Successful in small prog. only
  - → traced backward manually until location of cause
- Cause Elimination @ Involves Induction & Deduction

Induction : Specific to general

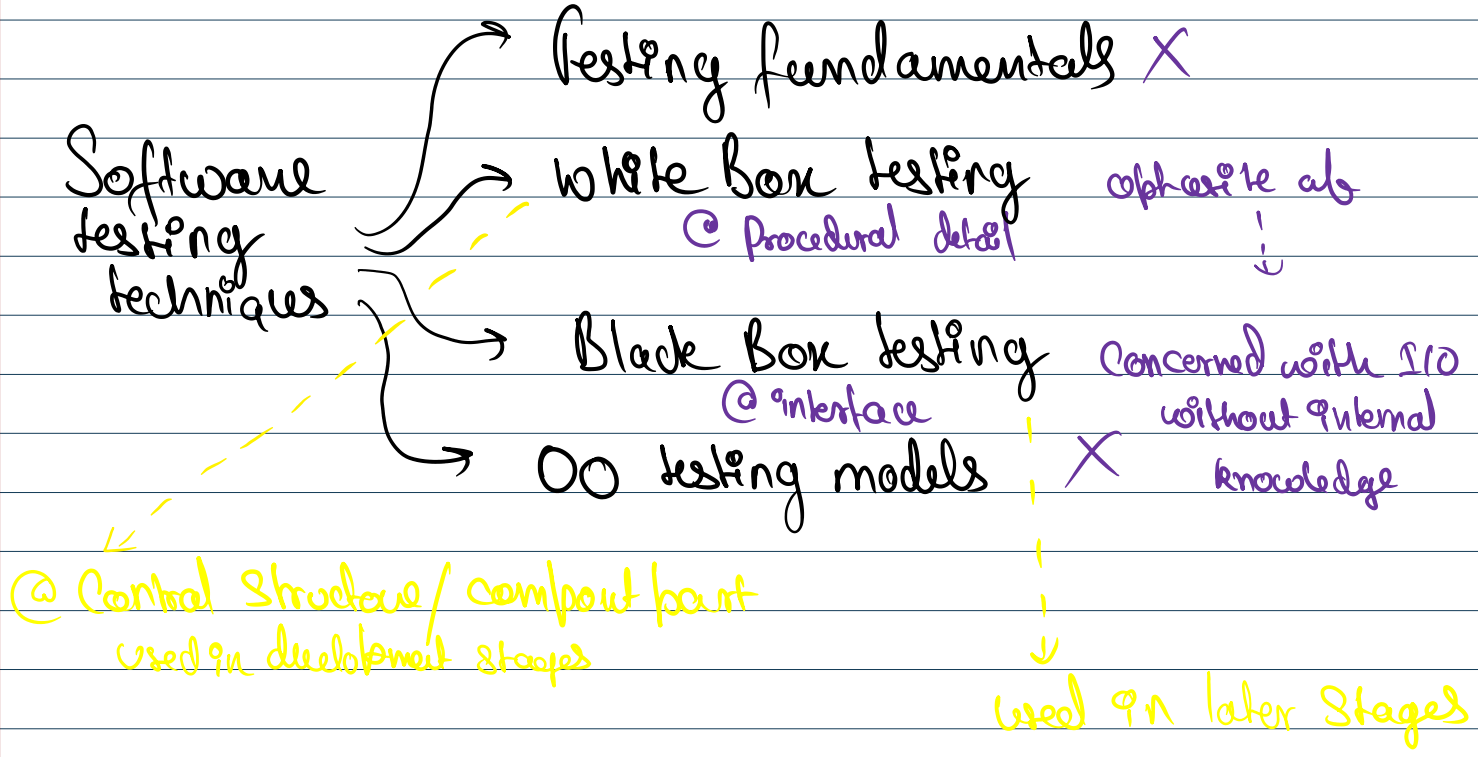
- Prove specific value is true then prove general case is true

Deduction : General to Specific

- Show that a specific conclusion follows from a set of general premises.

?Software Testing Techniques

28 December 2022 02:18 AM



!Flow Graph Notation & Cyclomatic Complexity + Testing of Loops

28 December 2022 02:24 AM

Complexity Number	Meaning
1-10	Structured and well written code High Testability Cost and Effort is less
10-20	Complex Code Medium Testability Cost and effort is Medium
20-40	Very complex Code Low Testability Cost and Effort are high
>40	Not at all testable Very high Cost and Effort

!!Equivalence Partitioning

28 December 2022 02:27 AM

Characteristics of Testable Software and Tests

28 December 2022 02:32 AM

Characteristics of testable Software

- Operable
- Observable
- Controllable
- Decomposeable
- Simple
- Stable
- Understandable

Characteristics of test

- A good test has high probability of finding error
- It is not redundant
- Should be best of breed
- neither too simple nor to complex