SOFTWARE ENGINEERING

Functionality
Localizability
Usability
Reliability
Performance
Security
+ Portability, Efficiency, Maintenance.

① Software Project lifecycle SPLC

1. Entry Criteria
2. Task & Deliverables
3. Exit Criteria.

Who | Dependencies | constraints

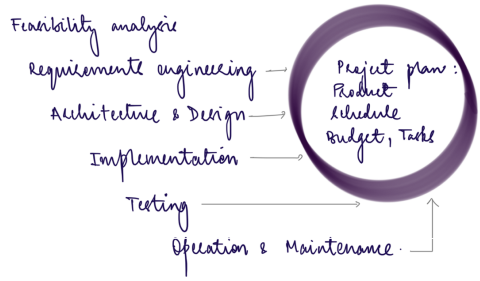
② Software Development life cycle SDLC

Requirement Analysis
Design
Implementation
Testing
Maintenance.

③ Product Development life cycle PDLC

Brainstorm Define Design Test Launch.

④ Project Management life cycle PMLC



Initiation, planning — Project execution — closure, project review
duration & control process improvement

⑤ Software Maintenance life cycle SMLC

PI 1. Problem / Task identification
A 2. Analysis : Impact, Criticality, Priority
A 3. Approval for change
DIR 4. Design | Implement | Review change
T 5. System regression
Acceptance testing
D 6. Delivery (Patches, Releases)

⑥ Product Lifecycle PLC

1. INTRODUCTION - Advertising, Marketing campaigns
2. GROWTH - growing demand, production increase expansion in availability
3. MATURITY - profitable stage
4. DECLINE - lower market share.

DISCONTINUANCE
OBSOLESCENCE

LEGACY SOFTWARE DEVELOPEMENT LIFE CYCLES

1. Waterfall Model

Pros	Cons
• Simple • Clear identified phases • Easy to manage • Specific deliverables • Easy to departmentalize + control	• Frozen requirements • Sequential • Big bang approach • High risk + uncertainty

2. V Model

3. Prototype model

4. Incremental model : Requirements are partitioned

5. Iterative model : starts from skeleton

Incremental model	Iterative Model
- Revisit, refine everything	• No revisiting
• Focus on details	• Focus on implementation
- Leverage on learnings	• Doesn't leverage learnings

AGILE

Rapid Iterative Co-operative Quality driven Adaptable

- Customer involvement
- Incremental delivery
- People not process
- Embrace change
- Maintain simplicity

SCRUM — Framework of rules, roles, events & artifacts

— Iterative approach, consists of sprints

- Product backlog
- sprint backlog
- Burndown chart

- Product Owner
- Scrum master
- Team

- Sprint Planning
- Daily scrum
- Sprint Review

CBSE - Component Based Software Engineering

Reuse based approach to select off-the-shelf components and integrate loosely coupled independent components into systems.

1. Standardized — conform to a std. model
2. Independent — Deploy w/o dependencies
3. Composable — All external interactions take place through publicly defined interfaces.

Feasibility Study

Project Identification
Feasibility Study
Feasibility Report
Approval

→ Requirements engineering

Requirements Engineering

E Elicitation – Active | Passive
A Analysis – MoSCoW, Modelling, Fishbone, UML
S Specification – SRS
V Validation – RTM [Requirement Traceability Matrix]
M Management – Change control, Traceability, Version control, Status tracking

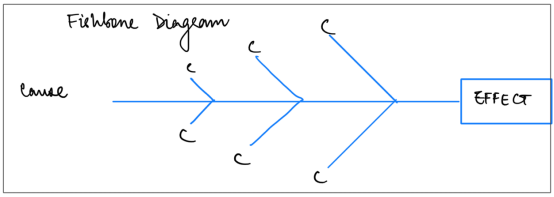
Function Requirements

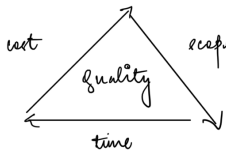
Non functional Requirements

User Requirements

System requirements

Domain Requirements





project management triangle

Software Project Management Lifecycle

1. Project initiation and approval
   - Project charter
   - Project owner
   - Initial budget
   - Identification of resources
2. Project planning [WBS, Allocating resources, risks]
3. Monitoring and control [Quantitative data]
4. Project closure

CoCoMo – Cost constructive Model

Regression model based on LOC.

- Effort – Amount of labour person-months
- Schedule – Amount of time required for completion of the job.

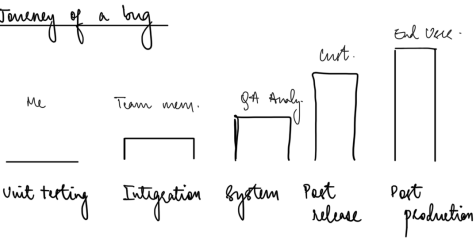
$E = a(KLOC)^b$

$time = c[Effort]^d$

Persons required = Effort / Time

- Organic
- Semi detached
- Embedded

Journey of a bug



Messy, verbose, cryptic, neat – Personalities

Naming conventions –

1. Pascal Casing

2. camelCasing
3. _underscoreBeforePrefix

Characteristics of Code –

1. Structure – Logically grouped.
well partitioned,
code & data structure

2. Readability – standardized indentation,
blank lines
space for blocks of code.

3. Comments

Coding Practices

(i) Defensive Programming

- Redundant code is incorporated to check system state
- Implicit assumptions tested explicitly.

(ii) Secure Coding

- Validate input
- Heed compiler warnings
- Use static and dynamic analysis tools
- Default deny
- Principle of least privilege
- Elevated permissions
- Sanitize data

(iii) Testable Coding

- Assertions
- Test points
- Scaffolding
- Hardware
- Test stubs
- Instrumenting
- Building test data sets

Definitions:

- Sprint Burndown
  Graphically communicated key metric for completion of work
- Team velocity metric
  Amount of software completed during a sprint
- Throughput
  Total value added work by team
- Cycle time
  Total time elapsed from the moment work is started till completion.

Activities ensuring Construction Quality

- Peer review
- Unit testing
- Test first
- Code stepping
- Pair programming
- Debugging
- Code inspections
- Static analysis

UNIT TESTING TOOLS

① Unit testing frameworks

- Enter method name, parameter, expected results
- NUnit (.NET) or JUnit (Java)

② Code coverage analyzers / Debuggers

- Identifies code not covered by analyzers.
  eg: JaCoCo / CoCo
  eg: GDB

③ Record playback tools

- Record keystrokes and mouse clicks
  eg: Selenium.

④ Wizards

· Generates code from input parameters.

Recap – Unit testing tools

- Unit testing frameworks [NUnit, JUnit]
- Code coverage analysers & debuggers [Code Tobobo]
- Record-playback tools
- Wizards

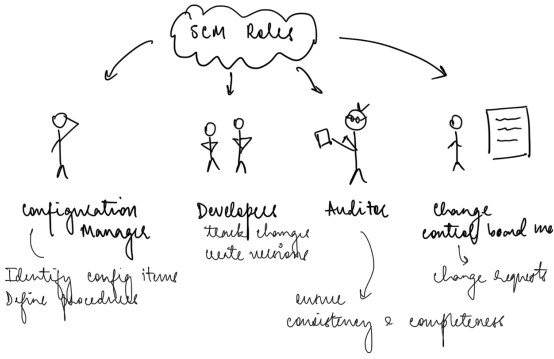
SCM

SC item identification – baseline components

SC control – deal w/ proposals

SC Auditing – to what degree does it reflect

SC status accounting – maintaining a record.



SCM Directories

1. Software Repository – Static library
2. Programmer's directory – Dynamic library
3. Master directory – Controlled library

SCM Management Tools

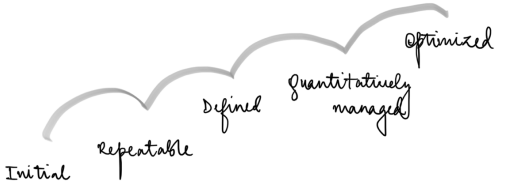
a) Source code Administration GitHub

b) Software build Maven

c) Software installation wise installer

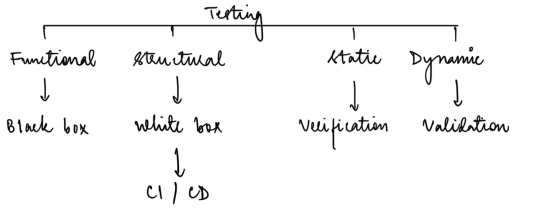
d) Software bug tracking Jenkins

Capability Maturity Model [CMM]



Verification – Software designed & developed as per specified requirements

Validation – Software (end product) has met the clients' true requirements



Static Analysis

1. Evaluation of code quality
2. Data flow
3. control flow
4. cyclomatic complexity

$M = E - N + 2P$

control flow graph: · Process oriented
· shows all the paths that can be traversed during a pro
· directed graph

Dynamic Testing

Code | Fault based — Approach — How testing? — Levels of test

1. control flow
2. Data flow
3. Error guessing
4. Fault seeding
5. Mutation testing

1. Manual
2. Automatic

Unit testing
Integration &
System &
Acceptance t

1. Equivalence partitioning
2. Boundary value analysis
3. Specification based
4. Intuition based
5. Usage based
6. Application domain.

LEVELS OF TESTING

1. Unit testing – Smallest individually executable unit
2. Integration testing – Big bang | Iterative
3. System testing – Verify compliance w/ SRS
   Smoke & sanity, Software performance, Destructive
   Functional & Non functional, Stress, Regression.
4. Acceptance testing – Installation testing
   Alpha/ Beta testing
   Performance testing

Risk – Probability of an unwanted incident

Test suite

Active — In progress — Completed

Software Test Metric Characteristics

1. Quantitative
2. Understandable
3. Applicability
4. Repeatable
5. Economical
6. Language independent

Software Test Metrics

1. SLOC
2. Fault Density : No of faults / size of prog.
3. MTBF
4. Failure rate
5. Defect Distribution : % of defects attributed to a specific phase.

6. Defect Leakage

MAINTENANCE

SW modification	Correction	Enhancement
Proactively	Preventive maintenance	Perfective maintenance
Reactively	Corrective maintenance	Adaptive maintenance

Global SW development

→ Characterized by: Geographic distance
Linguistic distance
Cultural distance
Temporal distance

Ethics

- Personal
- Professional
- Business

ITSM Key Processes

Availability Management
Problem management
Network Management
Storage Management
Configuration management
Facilities Management
Change management
Production Acceptance
Capacity planning
Business continuity
Performance
Security

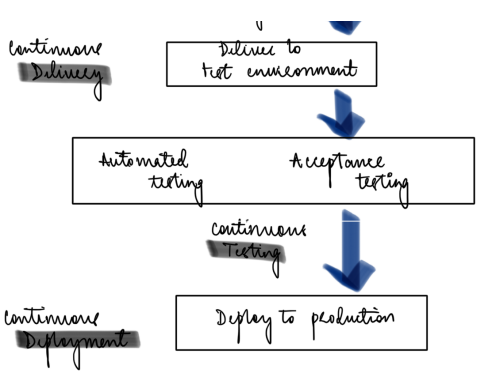
DEVOPS PIPELINE

Set of automated processes and tools that allows both developers and operations professionals to work cohesively to build and deploy code to a production environment.

Pillars of DevOps: CATS
Collaboration, Affinity, Tools, Scaling

Commit & Merge	Continuous build	Sanity testing

Continuous Integration


Continuous Delivery
Deliver to test environment
Automated testing
Acceptance testing
Continuous Testing
Continuous Deployment
Deploy to production