SOFTWARE
RE-ENGINEERING

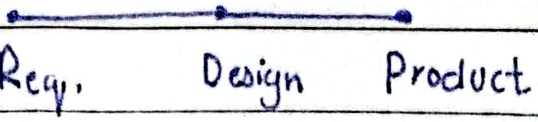
BSE : 7th
SEMESTER

DAY: ______

9/9/25

⊛ Software Engineering:

A process with some start and end involves different steps.



-> Changes w.r.t environment to make it scalable time by time.

-> Legacy system needs to be changed and do maintenance (It generally increases)

-> Poorly constructed system or product, so we wanted to reconstruct or move toward better.

-> Refactoring / Restructuring / Re-Evaluating
Re-Documenting

-> Reverse Engineering:
Extract from product, a code to redesign.

⊛ IDA Pro (Tools for Reverse Engineering)

11/9/25

⊛ Improves Code Quality and make documentation better.

16/9/25

⊛ Evolution of technology, we implement Software Re-Engineering.

⊙ **Reverse Engineering:**

Recreating missing things and not changing the main functionalities.

⊙ Re-Engineering

↙ Functionality change ↘ Technology change

⊙ **Legacy System:**

→ Any existing project difficult to maintain.

→ Characteristics:

⊙ Old systems

⊙ Maintainers problem

⊙ Large Projects + Inherited

⊙ Techniques Re-Engineering :

→ Automated analysis techniques .

→ Token Identification Analysis

→ Reverse Engineering by decompilation

: Syntax Decompiler/Analyzer

: Semantic Analyzer

⊙ Strategies in Re-Engineering:

⊙ Rewrite ⊙ Rework

⊙ Replace

18|09|25

⊙ Next Chapters Revise it and there will be talk about it. (Ch#2)

25|09|25

⊙ DevOps - 2009 introduced

→ **DevOps** → Operation Side (Production environment)

Development
(Implementation)

→ Dev Team → QA → Operations

→ In legacy system lag, time,

DATE:______ DAY:______

taking, blame game etc....). (Wait cycle more)

→ DevOps take over legacy system problems.

→ Tools and technologies use of DevOps to make system process better.

→ How the system works on other devices? (operations)

→ Jenkins tools (Pipelines)

→ QA & Testers participate from start of development.

→ If something is pushed over GitHub or pipeline, DevOps get notifications and start their working.

⊙ Main Terms:

⊙ CI | CD

CI → Continuous Integration

CD → Continuous Delivery/ Development / Depolyment

→ Steps:

(i) Code commit

(ii) Trigger build Pipeline

DATE: ________ DAY: ________

(iii) Build the code

(iv) Run automated Tests.

(c) Continuous Development / Delivery:

(i) Synchronization Automatic

(ii) Improvement / Maintained.

(Scope, Concepts, Practicality)

→ Jenkins, JDK, Git, Python

(Install till Next class.

14/10/25

→ Refactor, Rewrite, Restructure.

SDLC:

Software Development Lifecycle

→ Planning

→ Analysis (Requirements follows)

→ Designing (Implementation & Testing)

Decision Time:

→ Refactor

→ Rearchitect

→ Rewrite

→ Replace.

→ **Rewrite:**

Risks:

→ High bug count

→ Overheads

Benefits:

→ Testability.

Conditions.

→ Refactoring tried and failed

→ **Incremental Rewrite:**

Partial rewrite, some modules to rewrite.

DATE: DAY:

VS Code IDE:

- → Scripts for Jenkins file.
- → Jenkins Pipeline Lister Connection (Extensions)
- → Groovy plugin.

Declarative:

- → Agent available will work ("any" written)
- → Not wanted to run agent ("null" written)
- → Conditional ("when") derivative. (when equals, equals expected: 2, actual)
- → Parallel block will help us to run concurrently stages.
- → Fail Fast to stop the behavior
- → Parameters

Re-Architecturing:

Group of (different service & codebase same)

Higher level

- → Splitting a monolithic codebase into multiple components.
- → Collection of services.

DATE:

DAY:

⑦ Terminologics:

→ Monolithic codebase

→ module (or component)

→ Monolithic application.
(Runs on one machine)

→ API (Interfaces for call)

(SE) 9|10|25

⊙ Jenkins Master and Agent :

Master (Controls brains of Jenkins)

Agent (Execute the actual job assigned by master)

→ Categories :

⊙ Free Jobs (Single), Pipelines (Multi)

→ Add Slave

⊙ Node Add (Permanent and Temporary)

⊙ Pipeline :

→ Collection of jobs : Using code (Jenkins file)

→ Integration of pipeline.

⊛ Pipelines Types:

(i) Declarative Syntax

(ii) Descriptive/Scripted Syntax

⊙ Groory language.

⊙ Pipeline -> Agents -> Stages -> Work

⊛ Environment Variable:

->Global variable in a dictionary form.