

Lecture-1

Software Engineering

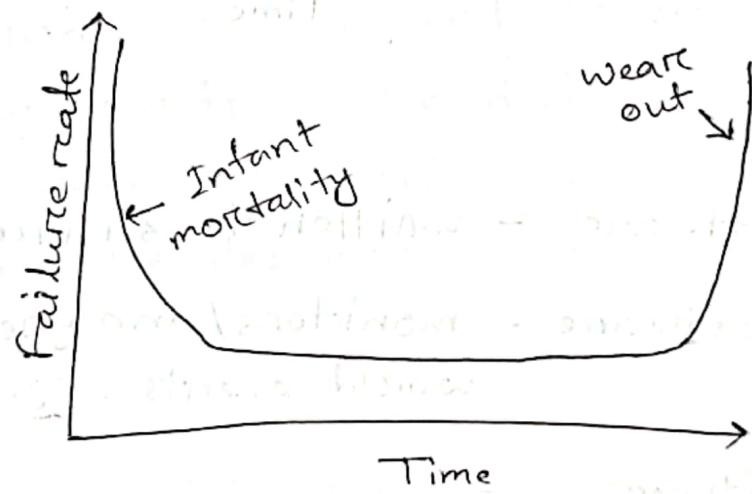
→ cost effective

→ Reliable

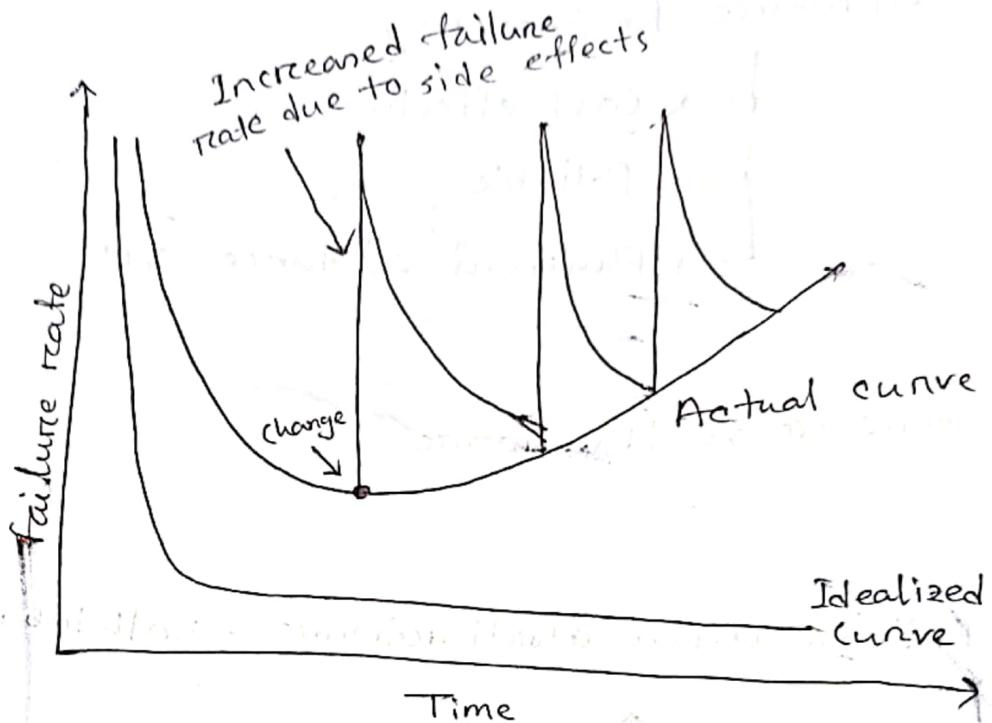
→ Physical existence \rightarrow IT

Software vs Hardware

Failure curve of Hardware - Bathtub curve



Idealized and actual failure curves for software



System Software :- written to service other programs

Realtime Software - monitors/analyzes/controls real-world events. e.g. Zoom

Business software -

Engineering and scientific software -

Embedded software - resides only in read-only memory

Personal computer software

Web-based software

Artificial Intelligence Software

Software Development Life cycle

i. Recognition of need / Requirement collection

- Initial investigation
- Getting the detail and precise requirement
- Performed by senior team members
- Defines the scope, objectives of the project.

ii. Feasibility study

- Software Requirement Specification (SRS) : Product requirements to be designed and developed throughout the life cycle. Everything which should be designed and developed.
- Technical : Whether the current computer system can support the new software
- Economic : Is the budget sufficient for developing the software?
- Legal : Cyber law and other regulations.
- Operation : Can operations be created which are expected by the client?
- Schedule : Can software be developed within the given time frame?

iii. Design

- Based on SRS, usually more than one design approach is proposed and documented in Design Document Specification (DDS).

- Describes the system from the software developer's viewpoint.
- System design

- Program design

* High Level Design (detail वर्तमान)

- Brief description and name of each module.
- Outline about the functionality of each module.
- Dependencies between modules
- Database tables with key elements.
- complete architecture diagram.

* Low Level Design (detail वर्तमान)

- Complete input, output of each module.
- functional logic of the modules.
- Database tables with type, size,
- Error messages.

iv. Implementation / Coding

- Whole task is divided into modules.
- Different modules are assigned to different developers.
- Developers write code to develop the module and thus the software.
- Raw code / framework
- Longest phase

v. Testing

- Verifying that the software works according to the customer requirement.
- Quality Assurance (QA) team and testing team tries to find bugs / defects.
- If they find so then they inform it to the developing team.
- Developing team solves the issue.
- Testing again by the QA and testing team until it is bug / defect-free and the quality standard defined in SRS is met.

vi. Installation / Deployment

- Releasing to a limited number of selected users.
- Releasing the final software.

vii. Maintenance

- Bug fixing : Bugs are reported by users and fixed.
- Update : Updating to newer versions.
- Enhancement : Adding new features.

Lecture-2

Chapter-2

Software Engineering :



- # I) Definition Phase - It focuses on what
- II) Development Phase - It focuses on how.
- III) Support Phase - It focuses on change associated with error correction, adaption, enhancement, prevention

Umbrella Activities :

- I. Software project tracking and control.
- II. Formal technical reviews.
- III. Software quality assurance.
- IV. Software configuration management.
- V. Document preparation and production
- VI. Reusability management.
- VII. Measurement
- VIII. Risk management.

Capability Maturity Model (CMM)

Level 1: Initial

Level 2: Repeatable

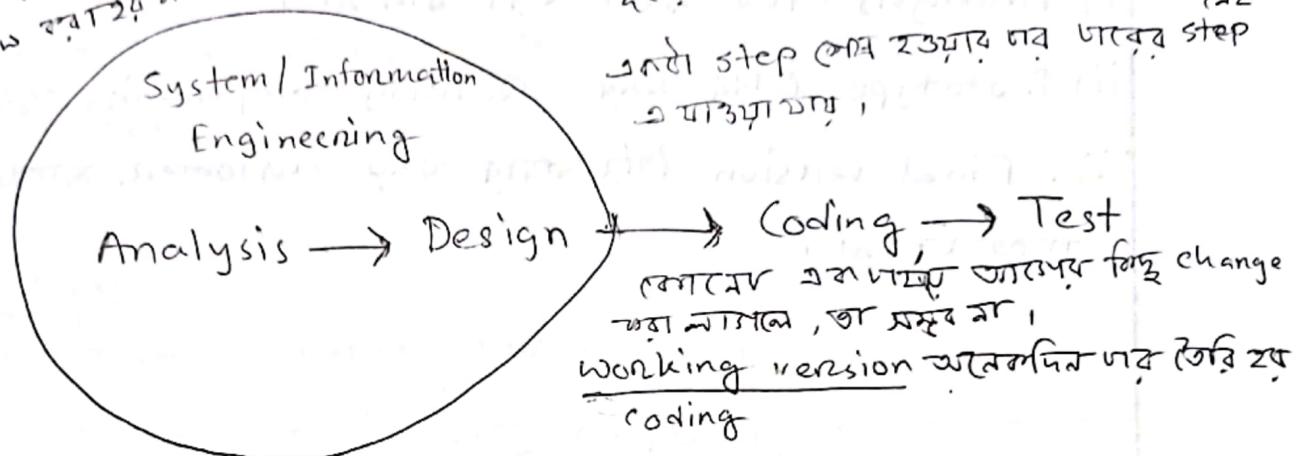
Level 3: Defined

Level 4: Managed ; level-3 तक पहुँचने वाले

Level 5: Optimizing

Software Process Models

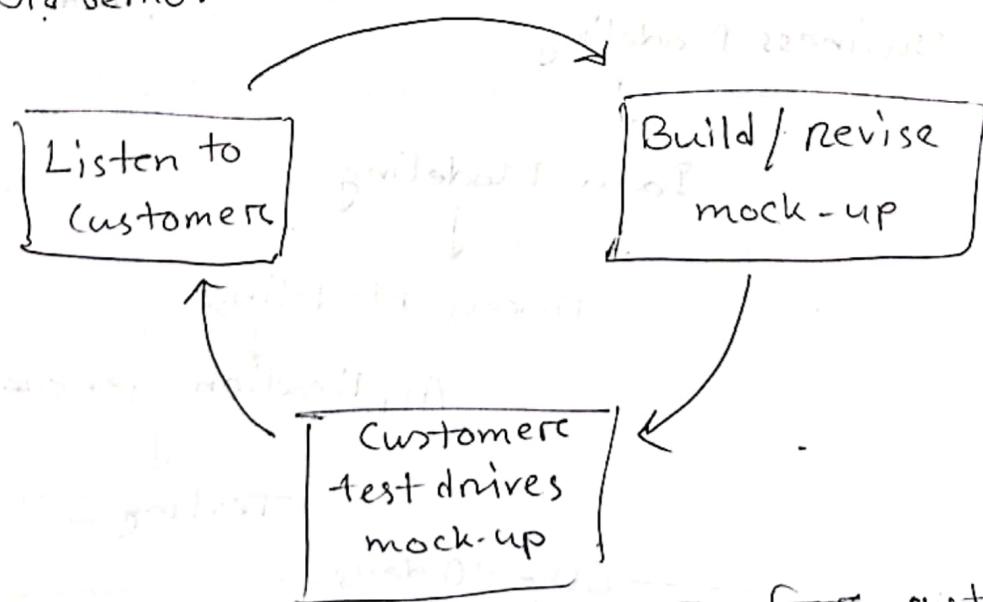
- i) Linear Sequential Model / Waterfall Model / Classic Life Cycle
Real project follow यहाँ तक कि.



इसी scenario को कौन सा चरण चाहता है, उसका जब process model match करें? क्या?

- ii) The Prototyping Model

Software as prototype - software as feature - इसका लाभ
- क्या है?



Prototype develop के बारे में customer के प्रिय, customer ने
change किया है.

Customer एवं साथी द्वारा बहुत बड़े बदलाव करके बदला देना।

Problems:

- i) Analysis फिर भी बदल देता है।
- ii) Prototype अतिक्रमित होता है। Quality compromise चलता है।
- iii) Final version लगभग असम्पूर्ण होता है। customer का उपयोग दिल्ले जाता है।
- iv) The RAD Model (Rapid Application Development)
उत्तराधिकारी ने बताया है कि इसमें component based construction.

इसमें निम्नलिखित RAD model follow किया जाता है।
अन्तर्गत team चाहते, विकास team वह product
Team #1 combine करके design
product बनाते हैं।

Business Modeling



Data Modeling



Process Modeling



Application generation



Testing & turnover

← 60 - 90 days →

Testing କଥା କାହାର କମ୍ପ୍ୟୁଟର ଦ୍ୱାରା କରାଯାଇଲା.

ଶୋଭା component model ମଧ୍ୟ ଏହାର ଫଳିତ କିମ୍ବା testing କରାଯାଇଲା.

Quickly କାହାର କମ୍ପ୍ୟୁଟର ଦ୍ୱାରା third party ହେଉ, ଅତିଶୀଘ୍ର କାହାର
feature ବିନ୍ଦୁ ଆନଳ୍ କରିବାକୁ test କରାଯାଇଲା କିମ୍ବା ନାହିଁ, ଆତିଥିର
ଫଳିତ tested.

Problems:

- i. Human resource କିମ୍ବା ଦୃଶ୍ୟକାରୀ,
- ii. କିମ୍ବା କାହାର କିମ୍ବା component ଏବଂ କାହାର କାହାର କାହାର
- iii. RAD model ଦିଲ୍ଲେ ହାବି ନାହିଁ.

iv. The Incremental Model

Linear sequential model + prototyping model.

LSM କରସାଧାରିତ କରିବାକୁ implement, then feedback କରିବାକୁ କାହାର

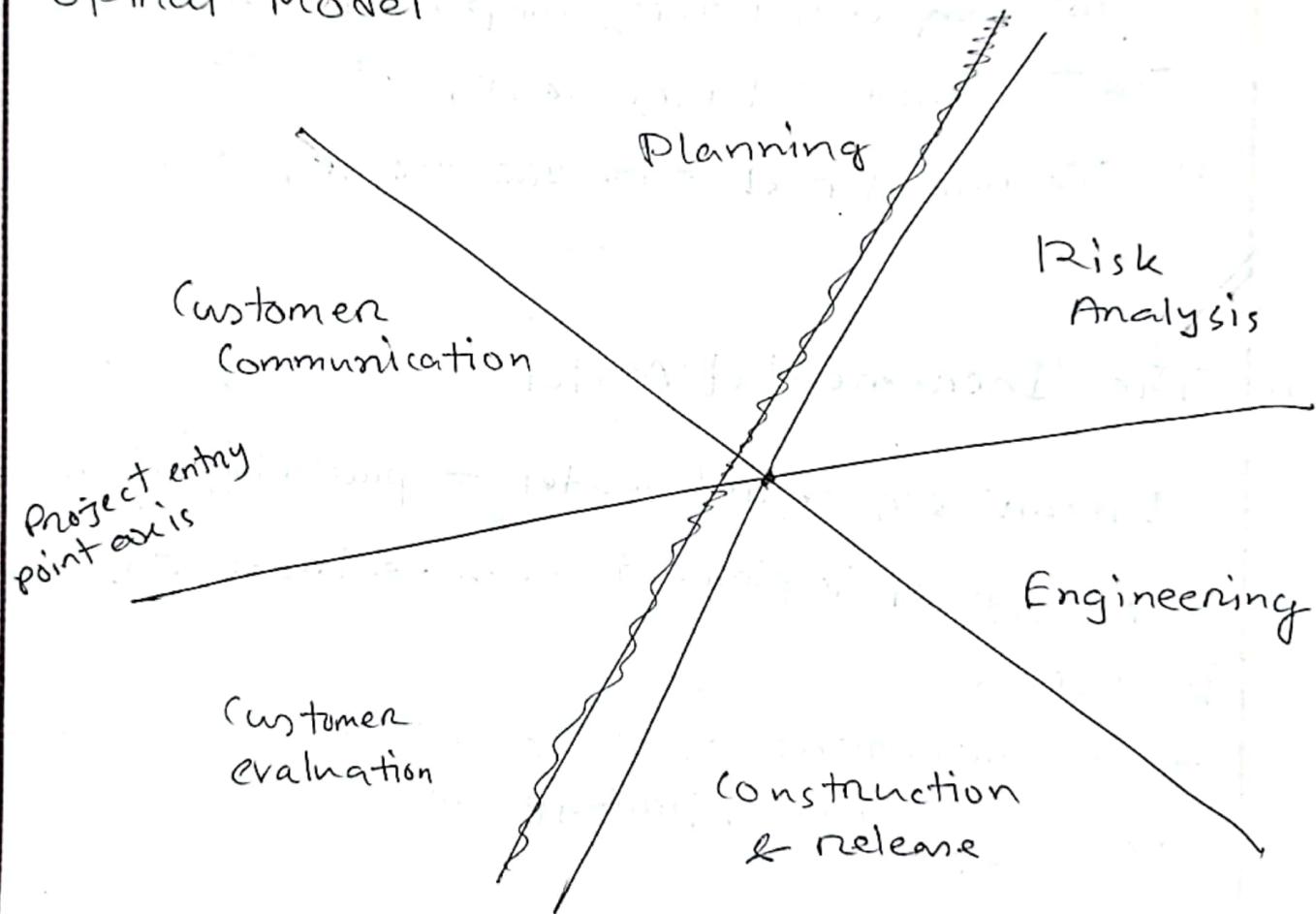
କିମ୍ବା LSM.

ପ୍ରତ୍ୟେକ increment କିମ୍ବା ଉପରେକୁ ଯାଇ client କେ ଫର୍ମରେ କାହାର
working version, prototype କାହାର.

Problems :

- I. ନିର୍ମାଣ ପାଇବା
- II. ଅବଶ୍ୟକ କମିଂ , duration କମି ହୋଇ , ଯେବେ କମିଂ
- III. Design phase ଯତକଣ୍ଠାୟ ଯାଏଥେ କାବାତ ହେଲା
ଏହି ଡିଜାଇନ ଏକାଗ୍ରାବେ ଥିଲା ତାହା ଯାଇ ଯାଇ ପରିବର୍ତ୍ତନ
modify କରାଯାଇ ।

V) Spiral Model



ଆରମ୍ଭାବୀ release ହୁଏ ,

Iteration କାମର ଅନ୍ତରେ କାବେ କାବେ କାବେ କାବେ

ଖର୍ବ୍ୟ କାମ କରୁଥିଲୁବେ ହୁଏ ନା ,

complex scenario କାବେ କାବେ କାବେ କାବେ

Agile Methods

- i) Iterative + Incremental
- ii) No detailed system specification
- iii) Minimized design documentation
- iv) Series of increments
- v) End-users, stakeholders involved in specifying, evaluating each increment.
- vi) Expensive tool support

* Principles

- I. Satisfying the customer.
- II. Welcoming changing requirements.
- III. Delivering working software frequently.
- IV. Frequent interaction with stakeholders.
- V. Maintaining constant pace.
- VI. Keeping it simple.
- VII. Building projects around motivated individuals.
- VIII. Empowering self-organizing teams.
- IX. Promoting sustainable development.
- X. Minimizing unnecessary works.
- XI. Reflecting and adjusting continuously.

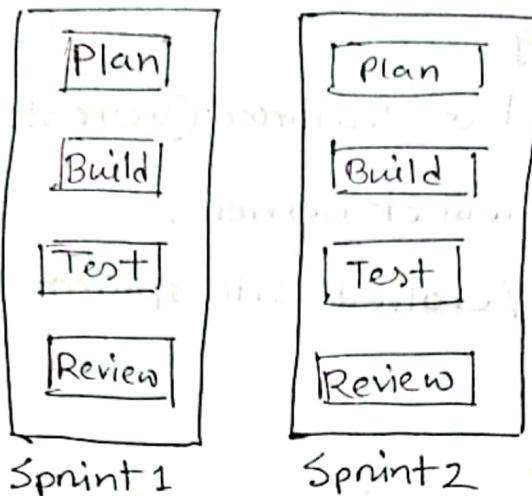
* Frameworks

- I. Scrum (गोपनीयता वाले उपयोग करते)
- II. XP
- III. Crystal
- IV. Kanban
- V. Lean.

Scrum

* Sprint:

1 to 3 weeks



एक स्प्रिंट डिलीवरी - Project sprint delivery

उत्पाद का एक प्रोडक्ट - One product

- Potentially Shippable Product

उत्पाद का एक बदलने वाला है - One changeable thing

* Roles :

I) Product owner - उत्पाद में प्रबंधक - Product manager

- Defining the product features
- Ideas comes from the customer

II) Scrum master

- Leader of the team
- Protecting the team, process
- Running meetings.

III) Team

- Helps developing products (developers, testers)
- Multiple roles.

* Documents

I) Product Backlog

- Prioritized list of features (user stories)
- Created by product owner.
- Gets changed / evolved with sprint.

II) User Stories

- Describes feature sets.

III) Sprint backlog

- Highest priority user stories.

IV) Burndown charts

- Progress during a sprint on the completion of sprint backlog.
- Should approach to 0 at the end.

* Ceremonies

I) Sprint planning

- Product owner, scrum master and team [meeting]
- Discussing user stories.
- Estimating relative sizes of sprints.

Sprint 1

II) Daily scrum

- Team [meeting]
- Accomplishment since last meeting.
- Current work.
- Blocked / need help.

III) Sprint review

- At the end of the sprint.
- Team demonstrates completed work to product owner.
- Discussion regarding progress.

Sprint 2

With business stakeholders

Product owner and stakeholders

Reviewing progress and discussing blockers.

Modifications

Planning next iteration

Identifying risks

Planning mitigation strategies

Planning next iteration

Identifying risks

Planning mitigation strategies

Lecture-3

Chapter-3 (Project Management Concept)

4P's - People, product, process, project

* The People - who works to build project.

The product - to understand objectives and scopes.

The process - comprehensive plan for software development.

The project - failure rate should be low.

The goal has to be fulfilled.

* The People :

* The Players :- categorized ^{into} by five

1. Senior manager
 2. Project manager
 3. Practitioners
 4. Customers
 5. End users
- } ^{***} difference between senior
and project manager
- } ^{***} differences .

* Team Leaders

1. Motivation

(MOI Model)

2. Organization

3. Ideas or innovation



* Project Manager characteristics

1. Problem solving

2. managerial identity

3. Achievement

4. Influence

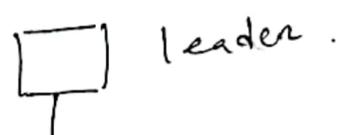
* The Software Team

1. Democratic Decentralized (DD)

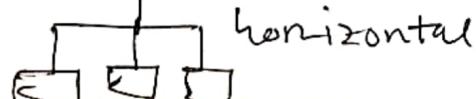


controlled way

2. Controlled Decentralized (CD)

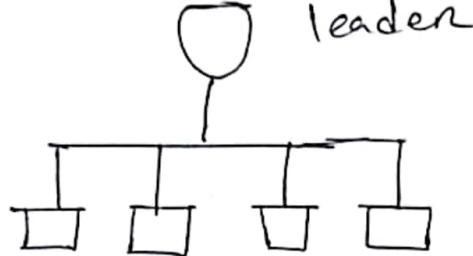


vertical



horizontal

3. Controlled Centralized (cc)



vertically
completes task faster, but efficient for simple problems
DD is best for difficult problems.

Seven Project factors

Coordination and communication Issues

* formal, Impersonal approach

* formal, Interpersonal procedure.

* Informal, Interpersonal procedure.

* Electronic communication

* Interpersonal networking

The Product

* Software Scope

1. context

2. Information objectives

3. function and performance

* Problem Decomposition

Divide and conquer strategy

The Process.

scenarios (what), which process model and why

(what needs to be done)

which team and why.

* Melding the product and the process.

Common Process framework -

for small project, customer communication activity?

for complex project, customer communication activity?

The Project

* five part common sense approach -

1. Start on the right foot.

2. Maintain Momentum.

3. Track progress.

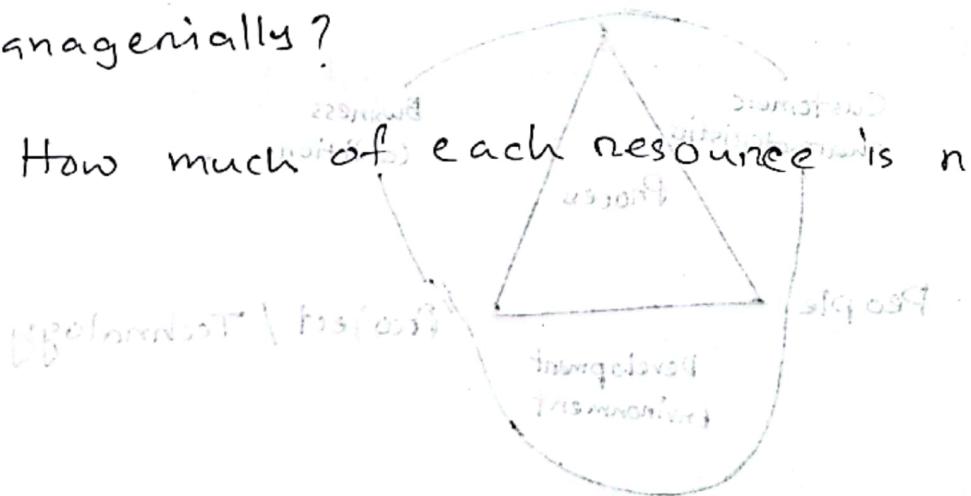
4. Make smart decisions.

5. Conduct a postmortem analysis.

to review what went wrong, what worked well

The W⁵HH Principle

1. Why is the system being developed?
2. What will be done, by when?
3. Who is responsible for a function?
4. Where are they organizationally located?
5. How will the job be done technically and managerially?
6. How much of each resource is needed?



Lecture-4 (Software Process and Project Metrics)

Chapter-4 (Software Process and Project Metrics)

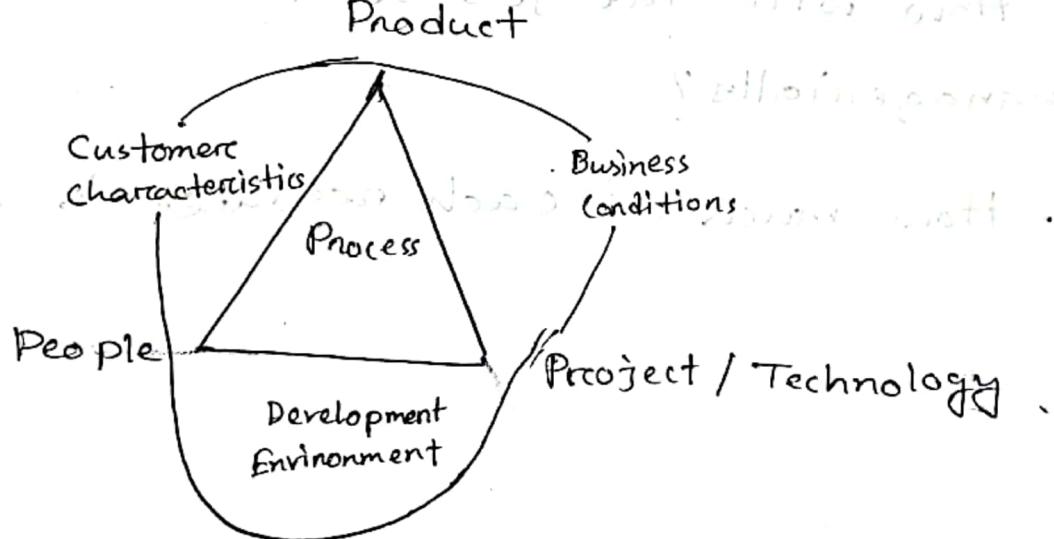
1 set question

Quantitative measurement → Direct

Qualitative → Indirect

Software Metrics (Measures project performance).

Business factors such as staff size, time spent, etc.



Software Measurement

• Definition - 2013

Direct Measure - cost and effort, lines of code, execution speed, memory size, defects

Indirect Measure - functionality, quality, complexity, efficiency, reliability, maintainability

• Indirect measure taking software

Indirect Measure - functionality, quality, complexity, efficiency, reliability, maintainability

Differences between direct and indirect metric measurement

• Direct measure is easier to understand

Size Oriented Metrics

Project	LOC	Effort	Cost \$/hr	Pp. doc.	Errors	Defects	People
Open source	1000	100 hours 10 days Person-month	100	Per hour	Product release Per day Per person-month	Project release Per day Per time error	Release Per day Per defect
Project X	1000	100 hours 10 days Person-month	100	Per hour	Product release Per day Per person-month	Project release Per day Per time error	Release Per day Per defect

size-oriented metrics universally accepted, LOC, LOC change etc.

Function-Oriented Metrics

function point calculation.

Weighting factor.

Measurement Parameter Count Simple Average Complex

Number of user input $10 \times \frac{3}{6} = 4.5$

Number of user output $15 \times \frac{4}{6} = 10$

Number of user inquiries $20 \times \frac{3}{6} = 10$

Number of files $30 \times \frac{7}{10} = 21$

Number of external interfaces $5 \times \frac{10}{10} = 5$

Count total $\rightarrow 44$

function Point,

$$FP = \text{count total} + [0.65 + 0.01 \times \sum (f_i)],$$

$f_i \in (i = 1 \text{ to } 19)$ complexity adjustment values.

question എണ്ണാം 0 to 5 എം മറ്റൊരു specify ആണ്

PJ Value പുല്ലാർ ഫി ഉം അംഗത്വം ഇവും

14 ടു് question മന ശാഖയിൽ ആണ്

Ex: 130 page

project ഏം description അനുസരിച്ച് question എണ്ണാം 440 ആണ്

$$FP = 440 \times [0.65 + 0.01 \times 34]$$

$$= 435.6$$

>

Measuring Quality

* Correctness

* Maintainability

* Integrity

* Usability

Chapter-10 (System Engineering)

Computer-Based Systems

1. Software

2. Hardware

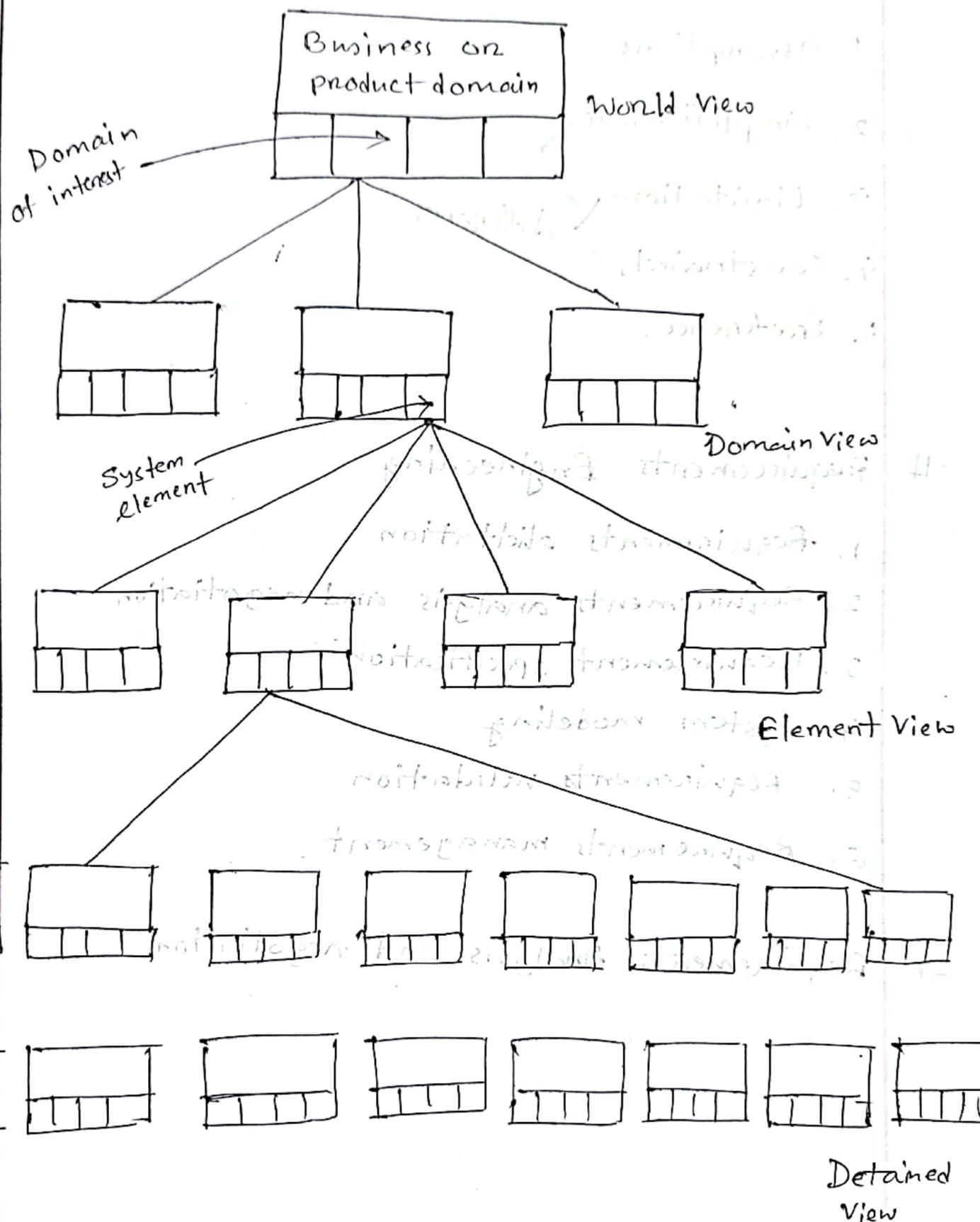
3. People

4. Database

5. Documentation

6. Procedures

System Engineering Hierarchy



System Modeling

1. Assumptions
2. Simplifications
3. Limitations
4. Constraints
5. Preference.

-# Requirements Engineering

1. Requirements elicitation
2. Requirements analysis and negotiation
3. Requirements specification
4. System modeling
5. Requirements validation
6. Requirements management.

Requirements Analysis and negotiation

Lecture - 5

Software Design

General definition of design :

The process of applying various techniques and principles for the purpose of defining a device, a process, or a system in sufficient detail to permit its physical realization.

Goal :

To produce a model or representation that will later be built out suitable traits like as practical, smart solutions, methods, assumptions etc.

Major areas of concern :

I. Data design

II. Architecture design

III. Interfaces design

IV. Components design .

A software design is a meaningful engineering representation of some software product that is to be built.

- A design can be traced to the customer's requirements and can be assessed for quality against predefined criteria.
- During the design process, the software specifications are transformed into design models that describe the details of the data structures, system architecture, interface, and components.
- Each design product is reviewed for quality before moving to the next phase of software development.

Q) Why is Design necessary?

- This is the only phase in which the customer's requirements can be accurately translated into a finished software product or system.
- * Software design serves as the foundation for all software engineering steps, that follow regardless of which process model is being employed.
- Without a proper design we risk building an unstable system:
- * One that may be difficult to test.
- * One whose quality can not be assessed until late in the software process.

Design and Software Quality

* Quality

- Design is the place where quality is fostered in software engineering, otherwise bad design

- Design provides us with representation of software that can be assessed for quality.

- Design is the only way we can translate customers requirements into finished software product or system.

- Software design serves as the foundation for all the software engineering and software support steps.

characteristics of a Good Design

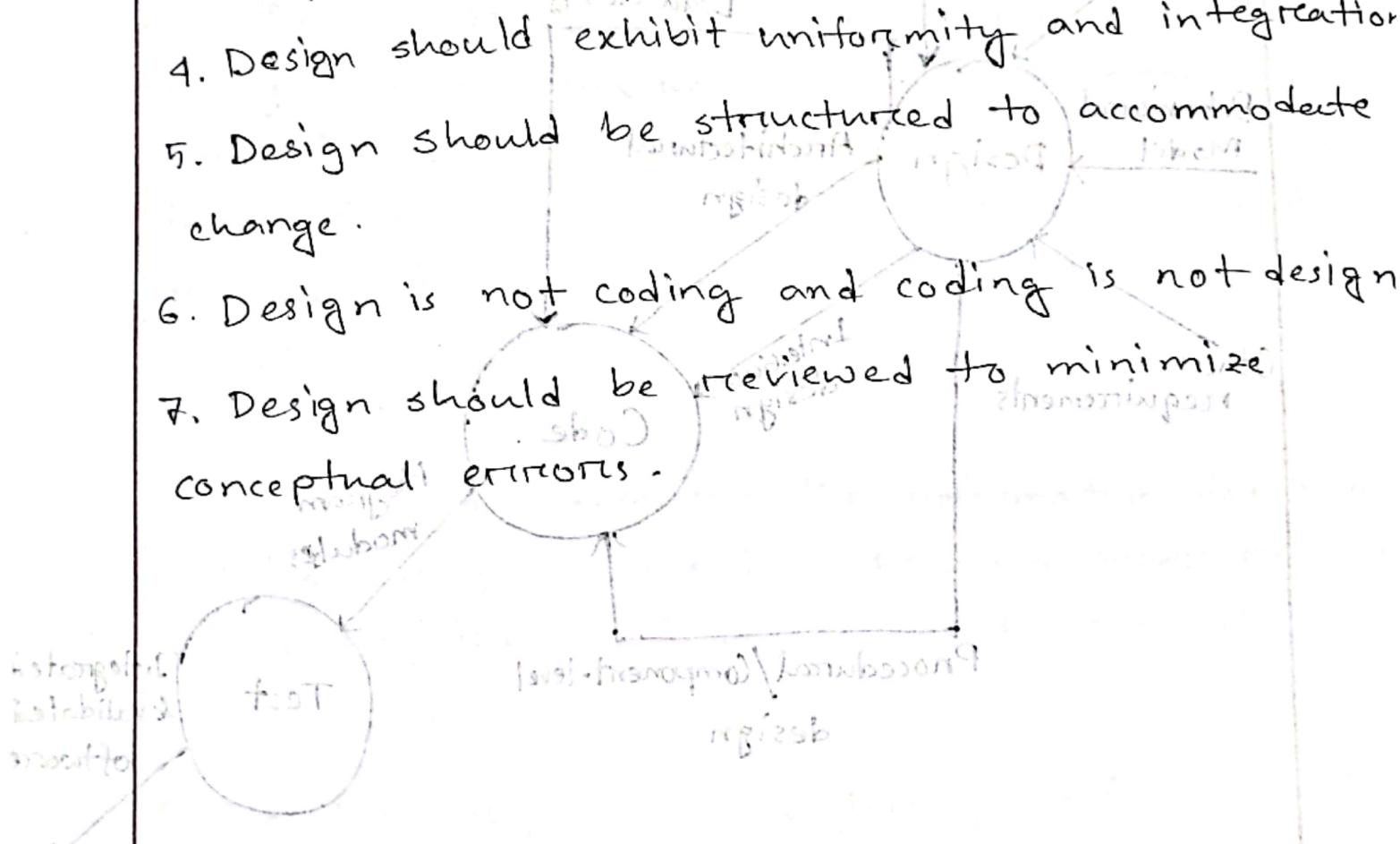
- The design must implement all of the explicit requirements contained in the analysis model, and it must accommodate all of the implicit requirements desired by the customer.
- The design must be readable, understandable guide for those who generate code and for those who test and support the software.
- The design should provide a complete picture of the software, addressing the data, functional and behavioral domains from an implementation perspective.

General Design Guidelines

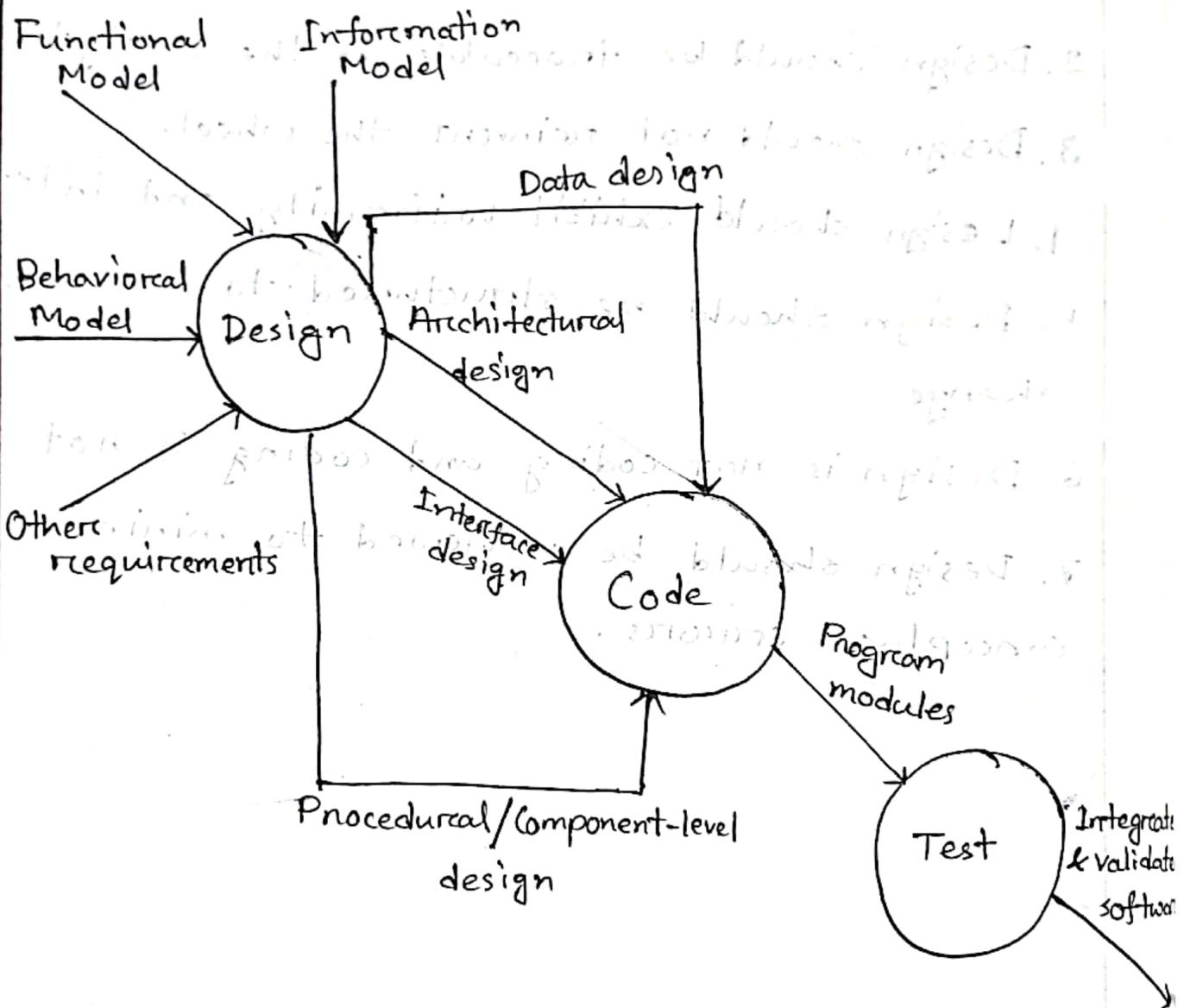
- A design should exhibit architectural structure that
 1. has been created using recognizable design patterns,
 2. is composed of components that exhibit good design characteristics,
 3. and can be implemented in an evolutionary fashion.
- Logically partitioned into components that perform specific tasks and subtasks.
- Lead to interfaces that reduce complexity of connections between modules and with the external environment.
- Derived using a repeatable method driven by information gathered during requirements.
- Lead to data structures that are appropriate for the objects to be implemented.

Design Principles

1. Design process should not suffer from tunnel vision
2. Design should be traceable to the analysis model.
3. Design should not reinvent the wheel.
4. Design should exhibit uniformity and integration
5. Design should be structured to accommodate change.
6. Design is not coding and coding is not design
7. Design should be reviewed to minimize conceptual errors.



Software Design Model



Data Design

Object Organization

The primary activity during data design is to select logical representation of data objects identified during the requirements definition and specification phase. The selection process may involve algorithmic analysis of alternative structures in order to determine the most efficient design or may simply involve the use of a set of modules that provide the operations upon some representation of an object.

Physical Organization

- * Data design transforms the information domain mode created during analysis into the data structures that will be required to implement the software.
- * The data objects and relationships defined in ERD (Entity Relationship Diagram) and the detailed data content depicted in the data dictionary provide the basis for the data design activities.

Architectural Design

- * Objective is to develop a modular program structure and represent the control relationships between modules.
- * Derived from the system specification, the analysis model, and the interaction of subsystems defined within the analysis model. It should be used to support design and synthesis activities.

Interface Design

- * Combines program and data structure by defining interfaces that allows data to flow throughout the program.
- * The interface design describes how the software communicates within itself, with systems that interoperate with it and with humans who use it.
- * Data and control flow diagrams provide information required for interface design.

Procedural Design

- * Transforms structural elements of the software architecture into a procedural description of software components.
- * After data & program structure have been established, it becomes necessary to specify procedure detail without ambiguity.
- * Information from process specification, control specification serve as the basis for component design.

Software Design Fundamentals

1. Abstraction
2. Refinement
3. Modularity
4. Software Architecture
5. Control Hierarchy
6. Data Structure
7. Software Procedure
8. Information Hiding

I. Abstraction

- Is one of the fundamental ways that can be used to cope with complexity in software development.
- Levels of detail / language used to describe a problem.
 - i. Highest level.
 - ii. Lower level.
 - iii. Lowest level.
- Types:
 - i. Procedural abstraction.
 - ii. Data abstraction.
 - iii. Control abstraction.

II. Refinement

- Top-down strategy
- Abstraction and Refinement are complementary.

III. Modularity

- Divide software into separate components often called modules that are integrated to solve problem requirements.

* Criteria to evaluate the design methods

- i. Modularity Decomposability
- ii. Modular Composability
- iii. Modular Understandability
- iv. Modular Continuity
- v. Modular Protection.

iv. Software Architecture

- The hierarchical structure of program components
the manner in which these components interact and
the structure of data that are used by the
components.

* Properties of an architectural design

i. Structural Properties:

Defines the components of the system, and
the manner in which those components are
packaged and interact with one another.

ii. Extra-functional properties:

Should address how the design architecture achieves requirements for performance, capacity, reliability, security, adaptability and other system characteristics.

iii. Families of related properties:

Identify the repeatable patterns that are commonly encountered in the design of similar systems. The design should have the ability to reuse architectural building blocks.

v. Control Hierarchy / Program Structure

- Organization of modules that implies a hierarchy of control.
- Depth, width, fan-out, fan-in
- Tree like control diagram.

vi. Data Structure

- Logical representation of the relationship among individual data elements
- Scalar, vector, array, linked list, stacks etc.

vii. Software Procedure

- Processing details of each module
- Precise specification includes sequence of events, decision points, repetitive operations, data organization.

viii. Information Hiding

- Modules should be "characterized by design decisions that each hides from all others."
- Modules are designed so that information within a module is inaccessible to other modules with no need for the information.
- Defines and enforces access information.

Modular Design

* Benefits

1. Reduces complexity.

2. Facilitates change.

3. Easier to develop.

4. Easier to maintain and test.

5. Easier implementation afforded by parallel development.

the program exhibits modularity, reusability, and maintainability.

Functional Independence

Design software so that each module addresses a specific sub-function of requirements and has a simple interface when viewed from other parts of the program structure.

* Benefits

1. Easier to develop.

2. Easier to maintain & test.

3. Measures

* Measures of Independence

1. Cohesion

- measure of relative functional strength of a module.

- a cohesive module should do just one thing / single task.

2. Coupling

- measure of the relative independence among modules.

The higher the score the stronger the coupling.

The lower the score the more independent the modules.

Score range from 0 to 100.

Score 0 = Strongly coupled modules.

Score 100 = Completely independent modules.

4. Pyramid



Lecture - 6

Chapter - 5

S/W Project Planning

(project ଏବଂ ଅନ୍ୟ କାହାର ଲୋକ-ଲାଭପାଇଁ ତଥା estimation)

Observations:

- complexity	- size	- Uncertainty
- Relative	- Interdependency	- Hierarchy(info.) - Compartmentalization(F*)
negotiation situations	out of control	suburbia process

- * কমপ্লেক্স ও আরও আরও মন্তব্য রয়েছে.
 - * quantitative estimation দেয়া মন্তব্য রয়েছে first এবং দ্বিতীয়ে,
 - but implementation এবং মান্তব্য ও দেয়া মন্তব্য,
 - * size আরও মন্তব্য coding এবং মান্তব্য,
 - * information কল্পনা দিতে হবে depending on interest.

Resource



1. Human resources	2. Reusable S/W Resources	3. Environmental S/W Resources
- Skills	- Off-the shelf components - full-exp. components - Partial-exp. components - New components	- tool / framework - support - Time window
Tech Organizational		

- * Off-the shelf components : project এর ফার্ম কম্পনি অন্তর্বর্তীর কোম্পানি থেকে পাওয়া যায়।
- * Commercial components : অন্তর্বর্তীর কোম্পানি থেকে পাওয়া যায়।
- * Full-exp. components : প্রজেক্ট এর জন্য কোম্পানি থেকে পাওয়া যায়। কোম্পানি থেকে পাওয়া যায়।
- * Partial-exp. components : অন্তর্বর্তীর কোম্পানি থেকে পাওয়া যায়। এবং এটি প্রজেক্ট এর জন্য সম্পূর্ণ কোম্পানি থেকে পাওয়া যায়।
- * New components : scratch করে প্রজেক্ট শুরু করা হয়।

first priority off the staff components (গোচর).
গান্ধি তার প্রেরণা করতে চাবত প্রয়োগ,

secondly full-exp. components try first, others team w/ experienced team ready support later.

3, 4 ଏହି ମଧ୍ୟ କିମ୍ବା situation depended, ତାରେ ଯେବେ ଯେବେ
optimally decide କରି,

जो नीचे दी हुई tool/frameworks लोड हवे, अगर कोत्रिमान्द्र अन्य सामाजिक ताक्षांक (एवं एन्सी) आमोदवा रखेगा support घाट / Helpdesk बोर्ड

COCOMO [Constructive Cost Model]

System Architecture	Size	Innovation	Deadline	Dev. Env.	KLOC
Organic	S	L	NT	St.	2-50
Semi-detached	M	M	M	M	51-300
Embedded	L	G	T	Cx	>300

S - Small
M - Medium
L - Large

G₁ - Great
T - Tight
NT - Not tight

C_x - Complex

S_t = Stable

KLOC - k-thousand lines
of codes

cocomo model ~~নথি কোড আসের পার্শ্ব রূপ কোড~~
classification ~~নথি কোড~~

KLOC ~~কোড দিচ্ছ যান্ত যান্ত~~, KLOC ~~কোড দিচ্ছ যান্ত যান্ত~~
factors ~~দুটি পার্শ্ব রূপ~~

Stages

- Basic
- Intermediate

Software risk factors ~~বোল্ড কোড স্টেবল~~ ~~TF~~

- Detail ~~স্টেবল কোড স্টেবল~~ ~~TF~~

$$MIFP = (0.05) \times S^2 C = 3$$

$$MIFP = (0.05) (S^2 C) \times MIF = 15$$

$$\frac{MIFP}{MIFP} = \frac{1}{15} = 0.06666666666666667$$

MIFP = 15

Lecture - 7

COCOMO

* Basic

$$\text{Effort} = a_1 (k\text{LOC})^{a_2} \xrightarrow{\text{Person-Month}} \text{PM}$$

$$\text{Time} = 2.5 (\text{Effort})^b M$$

$$\text{People} = \frac{\text{Effort}}{\text{Time}} P$$

a_1	a_2	b
0 → 2.4	1.05	0.38
5 → 3	1.12	0.35
E → 3.6	1.2	0.32

KLOC के अनुसार estimate करते हैं,

Consider an Embedded project with 700 kLOC. Calculate the Effort, development time, average staff size, productivity to develop the product.

$$E = 3.6 \times (700)^{1.2} = 9341.58 \text{ PM}$$

$$T = 2.5 \times (9341.58)^{0.32} = 46.61 \text{ M.}$$

$$\text{Average Staff Size} = \frac{E}{T} = \frac{9341.58}{46.61}$$

$$= 201 \text{ P.}$$

$$\text{Productivity} = \frac{k \text{LOC}}{E}$$

$$= \frac{700}{9341.57}$$

$$= 0.074 \text{ kLOC/PM}$$

Consider a project with approx. 7000 LOC.
Calculate the avg staff size, productivity.

$$E = 2.4 \times (7)^{1.05} = 24.793 \text{ PM} \approx 24.793 \text{ PM}$$

$$T = 2.5 \times (18.52)^{0.38} = 8.47 \text{ M} = 8.47 \text{ M}$$

$$P = \frac{E}{T} = \frac{24.793}{8.47} = 2.93 \text{ P} \approx \frac{18.52}{7.58} = 2.44 \text{ P}$$

$$\text{Productivity} = \frac{E}{T} \frac{\text{KLOC}}{E} = \frac{7}{18.52} = 0.38 \text{ KLOC/PM}$$

$$= 0.38 \text{ KLOC/PM}$$

* Intermediate

$$\text{Effort} = a_1 (\text{kLOC})^{a_2} \times \text{EAF} \text{ (Error Adjustment factor)}$$

$$\text{Time} = 2^{\frac{1}{b}} (\text{Effort})^{\frac{1}{b}}$$

$$\text{People} = \frac{\text{Effort}}{\text{Time}}$$

	a_1	a_2	b
0 → 3.2	1.05	0.38	
5 → 3.0	1.12	0.35	
E → 12.8	1.2	0.32	

	Very Low	Low	Nominal	High	Very High
Product Attributes	0.75	0.88	1	1.15	1.40
Required SW Reliability	0.75	0.94	1	1.08	1.16
Size of App. DB	0.75	0.85	1	1.15	1.3
Capacity of the Product	0.7	0.85	1	1.11	1.3
H/W Attributes	1	1	1	1.06	1.21
Runtime Performance Constraints	1	1	1	1.11	1.3
Memory Constraints	1	1	1	1.06	1.21
Volatility of Virtual Machine Environment	0.87	1	1.15	1.3	
Programming language Experience	0.94	1	1.07	1.15	

Lecture - 8

COCOMO

Intermediate

Cost Drivers

Cost Drivers

	VL	L	N	H	VH
<u>Personal Attributes</u>					
Analysis Capability	1.46	1.19	1	0.86	0.71
Application experience	1.29	1.13	1	0.91	0.82
S/W Engineering capability	1.42	1.17	1	0.86	0.70
Virtual Machine Experience	1.21	1.10	1	0.9	
Programming language exp.	1.14	1.07	1	0.85	
<u>Project Attributes</u>					
App & S/W Eng.	1.24	1.1	1	0.97	0.82
Use of S/W tools	1.24	1.1	1	0.91	0.83
Required Dev. Schedule	1.23	1.08	1	1.04	1.1

consider a project with an estimation of 250kLOC. Although the size of the App. DB is nominal, it has high memory constraints. The Tech. team has developers who have high app. experience but very low experience in programming. The required development schedule for the project is low. Calculate the effort, dev. time, avg. staff size required.

Soln:

Semi-detached

$$E = 3(250) \times (1 \times 1.06 \times 0.91 \times 1.14 \times 1.08)$$

$$\approx 1727.78 \text{ PM}$$

$$T = 2.5 (1727.78)^{0.35}$$

$$= 33.97 \text{ M}$$

$$P := \frac{E}{T} = \frac{1727.78}{33.97} = 50.86 \text{ P} \\ \approx 51 \text{ P.}$$

Lecture 9

Software Equation

$$E = \left[\frac{LOC \times B^{0.333}}{P} \right]^3 \times \frac{1}{t^1}$$

$$t = 8.14 \times \left(\frac{LOC}{P} \right)^{0.43}$$

$$E = 180 B t^3$$

→ in years

B [Special & skills Factor]

$$= 0.16 [5-15] K LOC$$

$$= 0.39 [7-10] K LOC$$

P [Productivity Parameter]

Real time
= 2000 [R.T Embedded S/W]

= 10000 [Telecommunication]

= 12000 [Scientific]

= 28000 [Business]

A scientific SW with approx. 75000 LOC is to be developed. Calculate the time, effort required for the SW.

$$B = 0.39$$

~~$$P = 12000$$~~

~~$$f = \frac{75000 \times (0.39)^{0.333}}{12000}$$~~

$$t = 8.14 \times \left(\frac{4.8}{\frac{75000}{12000}} \right)^{0.43} \approx 17.89 \text{ M}$$

$$E = 180 \times 0.39 \times \left(\frac{17.89}{12} \right)^3 \approx 233 \text{ PM}$$

$$\approx 233 \text{ PM}$$

A S/W using in the micro wave oven approx. 15000 LOC is to be developed. Calculate the time, effort.

$$B = 0.16$$

$$\rho = 2000 \text{ g}$$

$$t = 8.14 \times \left(\frac{15000}{2000} \right)^{0.43} \text{ M}$$

$$= 800.485 \text{ M}$$

$$= 19.36 \text{ M}$$

$$E = 180 \times 0.16 \times \left(\frac{19.36}{12} \right)^3$$

$$= 180 \times 0.16 \times 1.61^3$$

$$= 120.94 \text{ PM}$$

$$120.94 \times 60 = 7254 \text{ sec.} = T$$

~~for activity analysis~~

$$M = 120.94 \text{ sec.}$$

$$\frac{120.94}{120.94} = 1$$

$$\frac{120.94}{120.94} = 1$$

$$120.94 \times 1 = 120.94$$

$$120.94 \times 1 = 120.94$$

$$120.94 \times 1 = 120.94$$