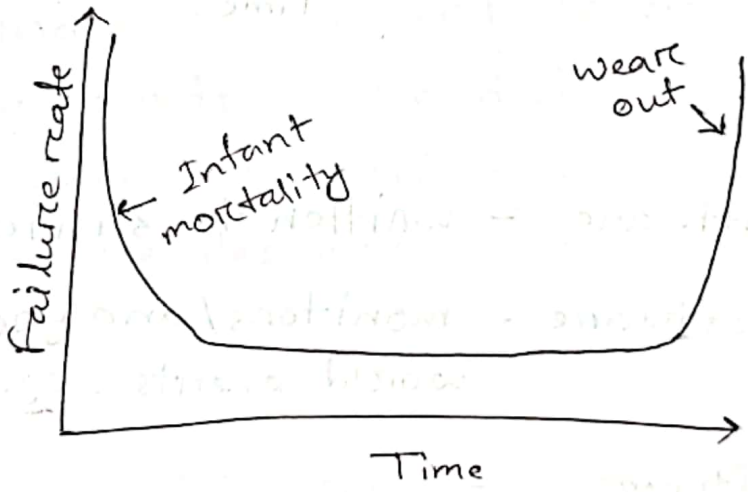
Lecture-1

Software Engineering

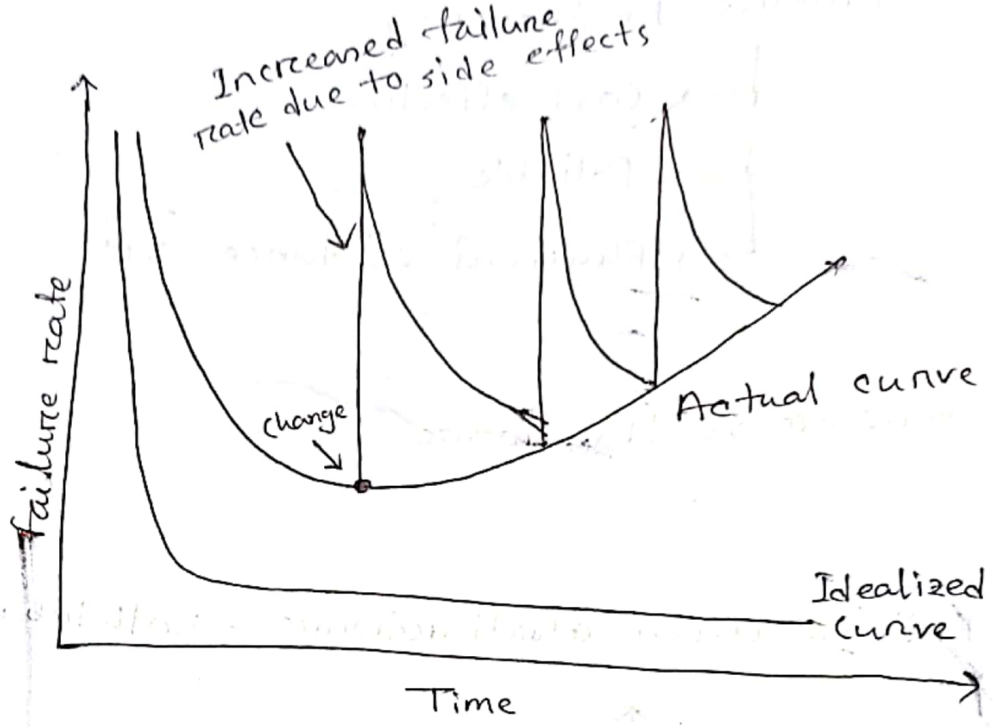
- cost effective
- Reliable
- Physical existance नहीं

Software vs Hardware

Failure curve of Hardware - Bathtub curve



Idealized and actual failure curves for software



System Software :- written to service other programs

Real time Software - monitors / analyzes / controls real-world events . eg. 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 expectedly 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 কম)
  - 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 বেশি)
  - 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 endusers.
- Releasing the final software.

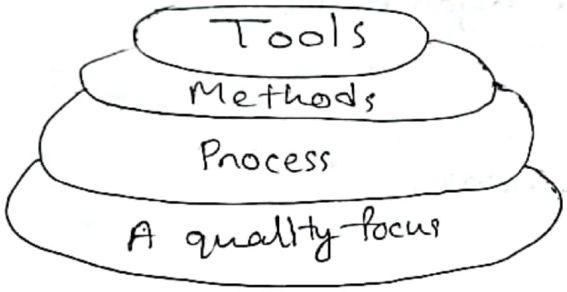
vii. Maintenance

- 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 এর সকল বৈশিষ্ট্য আছে.

Level 5: Optimizing.

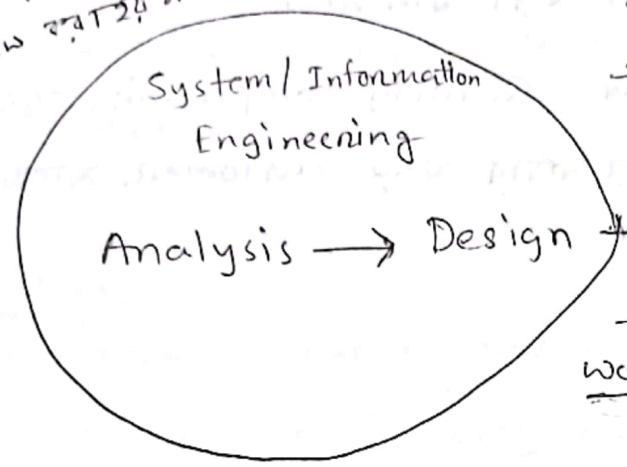
Software Process Models

i) Linear Sequential Model / Waterfall Model / Classic Life Cycle

Real project এ follow করা হয় না.

পূর্বের step এ ফিরে যাওয়ার সুযোগ নেই

একটা step শেষ হওয়ার পর পরের step এ যাওয়া যায়।



কোনো একটা অংশের কিছু change করা লাগলে, তা সম্ভব না।

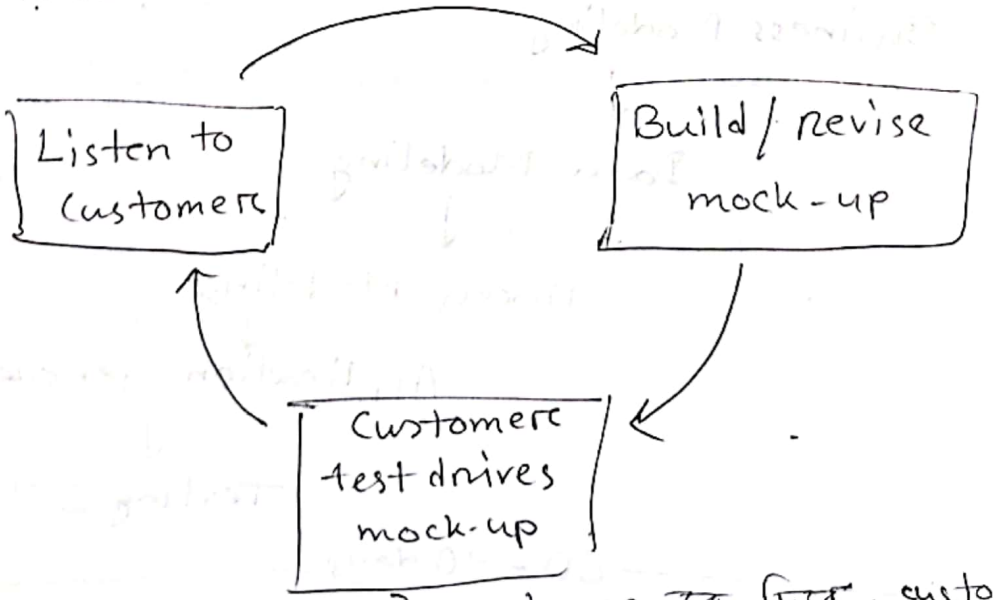
working version অনেকদিন পর তৈরি হয়

coding

একটা scenario বলে দেওয়া থাকবে, সেটার জন্য কোন process model match করে? কেন?

ii) The Prototyping Model

Software এর prototype - software এর feature কেমনে কাজ করবে তার demo.



Prototype develop করে এটা customer কে দিবে, customer এটা change করতে পারবে.

Customer এর কাছে থেকে বার বার ফ্রন্টে change করা যাবে.

Problems:

i) Analysis ঠিক মতো করা যায় না.

ii) Prototype তৈরির সময় Quality compromise চলে আসে.

iii) Final version তৈরি করার সময় customer আরো সময় দিতে চায় না.

iii) The RAD Model (Rapid Application Development)

খুবো কাজ ভাগ করা হয়.

component based construction.

সময় কম থাকলে RAD Model follow করা যায়.

অনেকগুলো team থাকবে, সবগুলো team এর product combine করে desire product পাওয়া যায়.

Team #1

Business Modeling

↓

Data Modeling

↓

Process Modeling

↓

Application generation

↓

Testing & turnover

← 60 - 90 days →

Testing এর কাজ কম হবে এখানে.

কোনো component model কিনে আনলে সেটা testing এ লাগবে না.

Quickly কাজ করার জন্য third party থেকে, অন্য কোনো feature কিনে আনলে সেটার test করার দরকার নাই, কারণ সেটা tested.

Problems:

i. অনেক human resource লাগবে,

ii. যে কোন কাজটা বিভিন্ন component এ ভাগ করা যায় না, সেটা RAD model দিয়ে হবে না.

iii. Technical risk বেশি হলে হবে না,

iv. The Incremental Model

Linear sequential model + prototyping model.

LSM অনুযায়ী implement, then feedback নিয়ে আবার LSM.

প্রত্যেক increment এর পরেই যেটা client কে দিবে সেটা working version, prototype না,

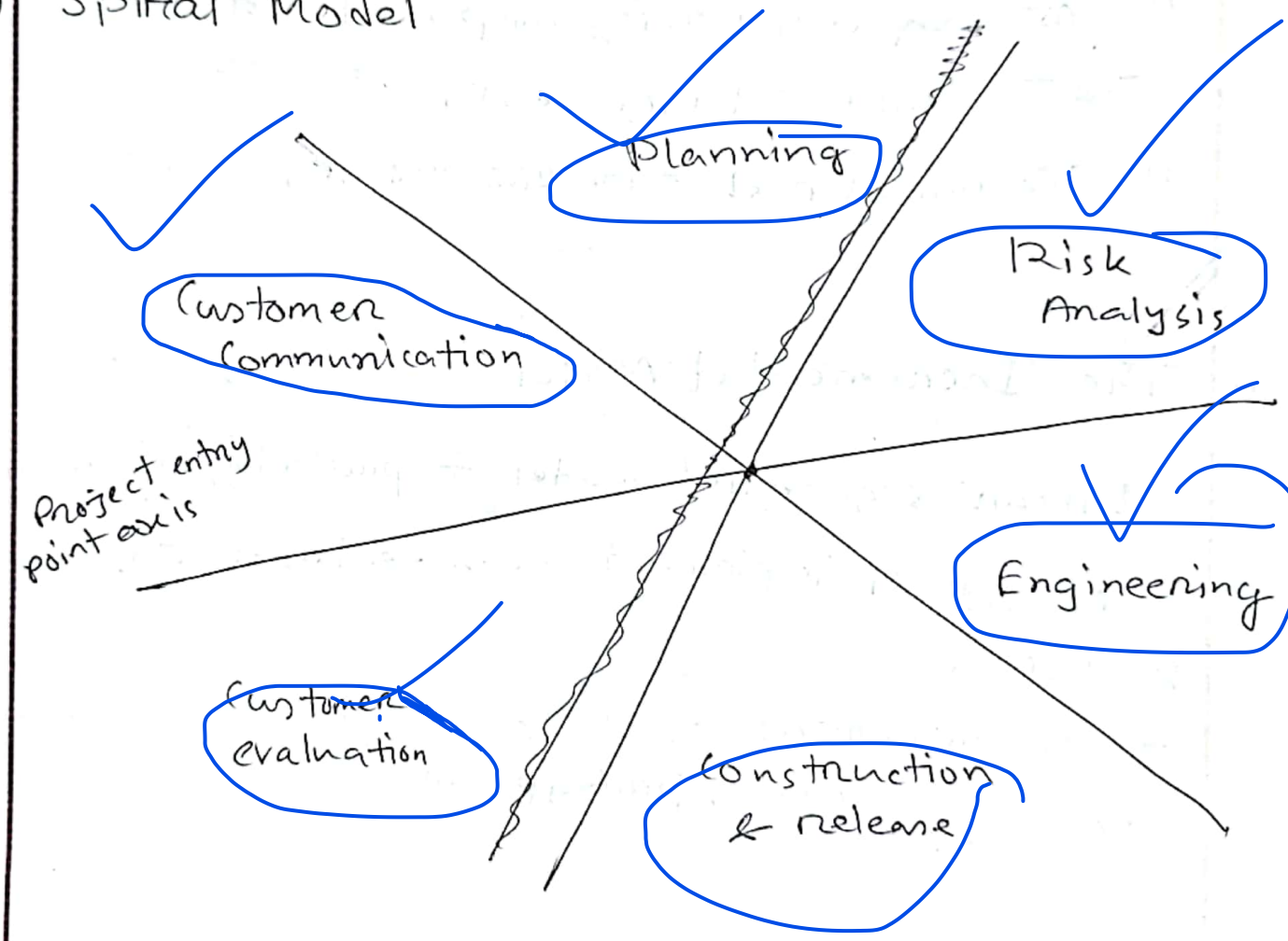
Problems :

i. সময় সাপেক্ষ

ii. খরচ বেশি ; duration বেশি হলে, খরচ বেশি.

iii. Design phase সতর্কতার সাথে করতে হবে।
২য় design এমনভাবে করতে হবে যেন পরবর্তীতে modify করা যায়।

b) Spiral Model



আনসফটনা release হয়।

Iteration কখন থামতে হবে তার নির্দিষ্ট সময় আগে থেকে থাকে না।

complex scenario তে ব্যবহার হয়।

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 customers
II. Welcoming changing requirements.
III. Delivering working software frequently.
IV. Frequent interaction with stake holders.
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

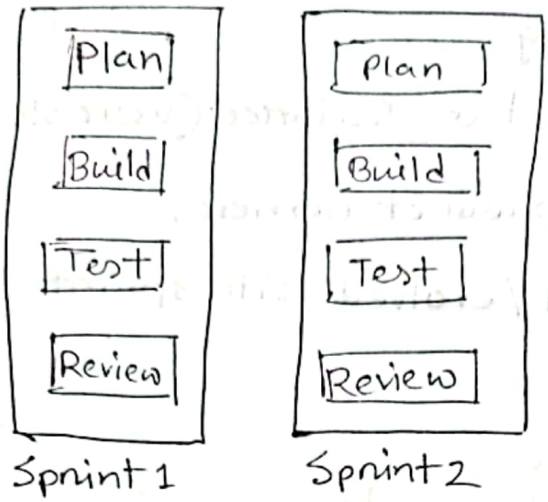
I. Scrum (৫০% এর বেশি use হয়)
II. XP
III. Crystal
IV. Kanban
V. Lean.

Scrum

Scrum

*Sprint:

1 to 3 weeks



Several incremental releases called Sprints.

প্রত্যেকটা sprint delivery হওয়ার মতো product - Potentially Shippable Product

* Roles :

1) Product owners
   - Defining the product features.
   - Ideas

11) Scrum masters
   - Leader of the team
   - Protecting the team, process
   - Running meetings.

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

* Documents

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

11) User Stories
   - Describes feature sets.

111) 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

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

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.

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 ~~by~~ into five

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

* Team Leaders

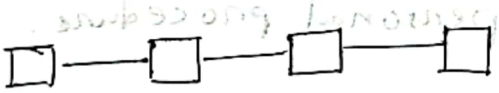
1. Motivation (MOI Model)
2. Organization
3. Ideas or innovation

* Project Manager characteristics:

1. Problem solving
2. managerial identity
3. Achievement
4. Influence and team building

* The Software Team

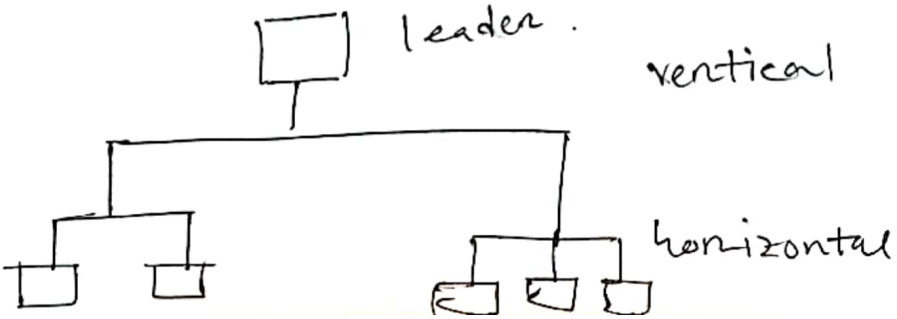
1. Democratic Decentralized (DD)



horizontal

Controlled way নাই.

2. Controlled Decentralized (CD)

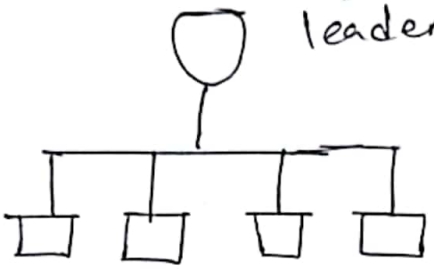


leader.

vertical

horizontal

3. Controlled Centralized (CC)



no sub-group

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.

scenario কোথায়কবে, ~~কোনটা~~ which process model and why

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?

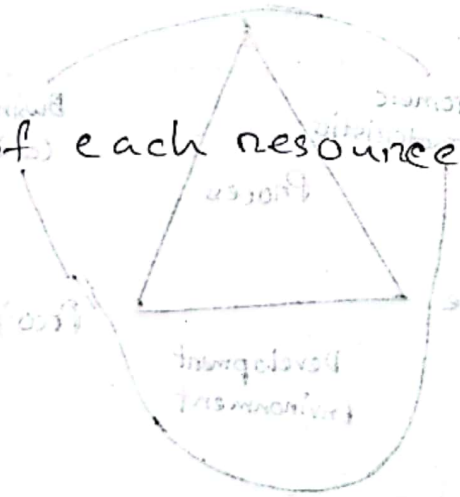
The Project

* 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.

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?



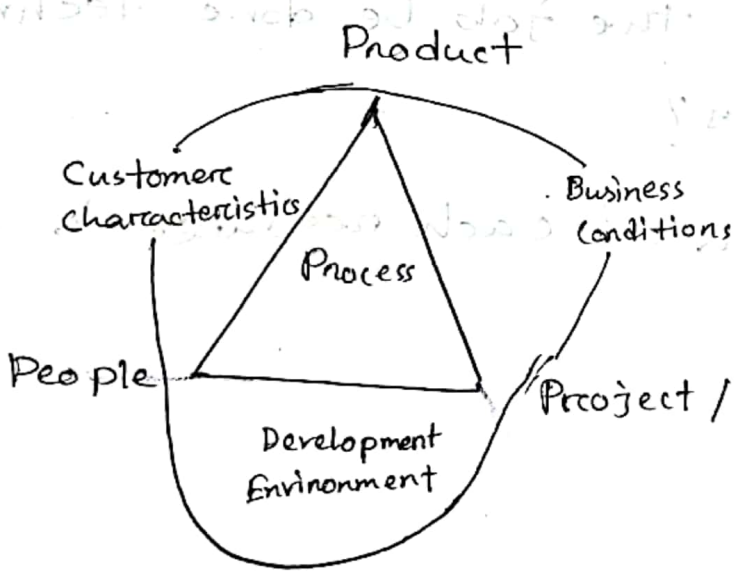
Lecture-4

Chapter-4 (Software Process and Project Metrics)

1 set question

Quantitive measurement → Direct

→ Indirect



Software Measurement

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

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

Differences between direct and indirect ~~metric~~ measurement

Size Oriented Metrics - Direct Measurement

Project	LOC	Effort কতজন মানুষ কতদিন একসাথে কাজ করেছে Person-month	Cost $(000)	Pg. doc. Per page	Errors Product release এর আগে যে পরিমাণ error	Defects Project release করার পর একটা time এর মধ্যে কতগুলো defect	People

Size-oriented metrics universally accepted হয় না.
LOC change হতে পারে.

Function-Oriented Metrics

function point calculation.

Measurement Parameter	Count		Weighting factor. Simple	Average	Complex	
Number of user input	10	×	3	4	6	= 30
Number of user output	15	×	4	5	7	= 105
Number of user inquiries	20	×	3	4	6	= 60
Number of files	30	×	7	10	15	= 240
Number of external interfaces	5	×	5	7	10	= 35
Count total	→					440

Function Point,

$$FP = \text{count total} + [0.65 + 0.01 \times \Sigma(F_i)].$$

F_i ∈ (i = 1 to 14) Complexity adjustment values.

question এর ans 0 to 5 এর মধ্যে specify করে
এর Value গুলো fi এ রাখতে হবে।
14 টা question মনে রাখতে হবে।

Ex: 130 page

project এর description অনুযায়ী question এর ans দিতে হবে।

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

$$= 435.6$$

Measuring Quality

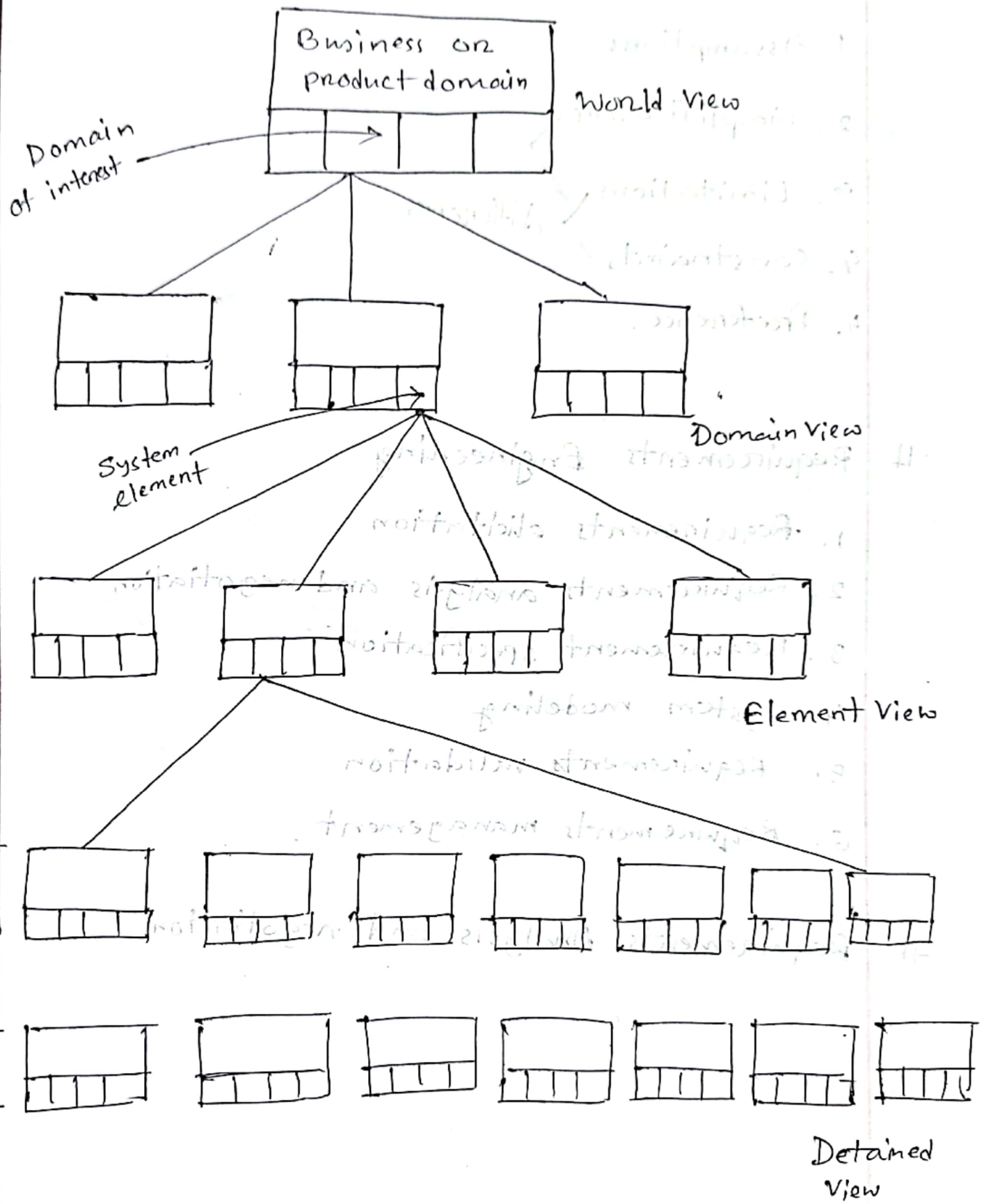
* Correctness
* Maintainability
* Integrity
* Usability

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

(2, 3, 4 → differences)

4. 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.

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 ~~sep~~ 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.

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.

- 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.

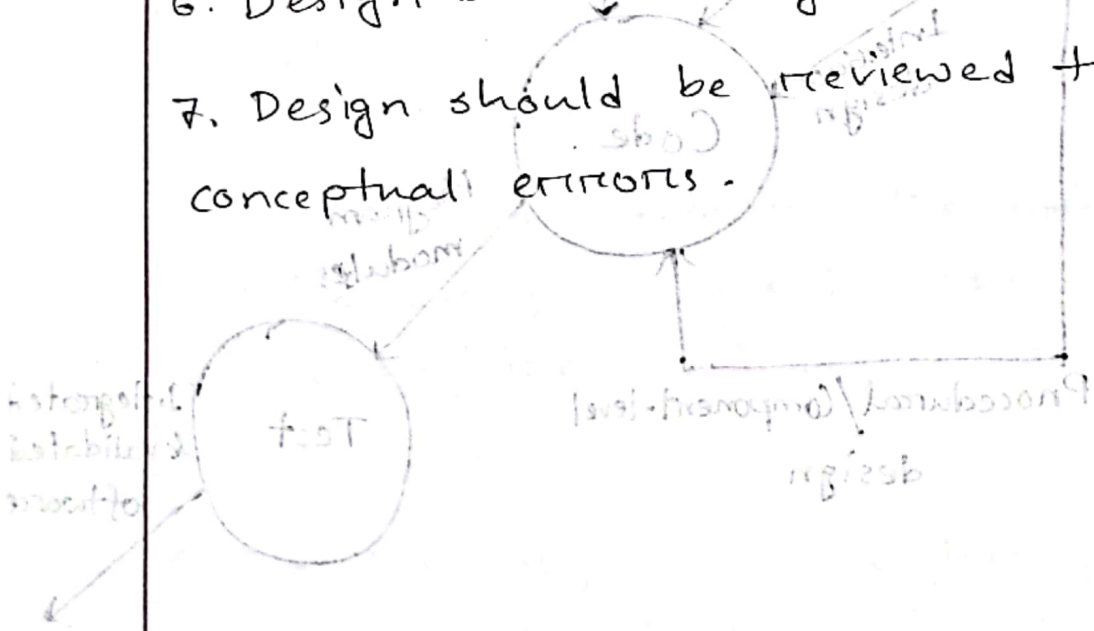
- Lead to interfaces that reduce complexity of connections between modules and with the external environment.

- Derived using a repeatable method driven by information gathered during requirements.

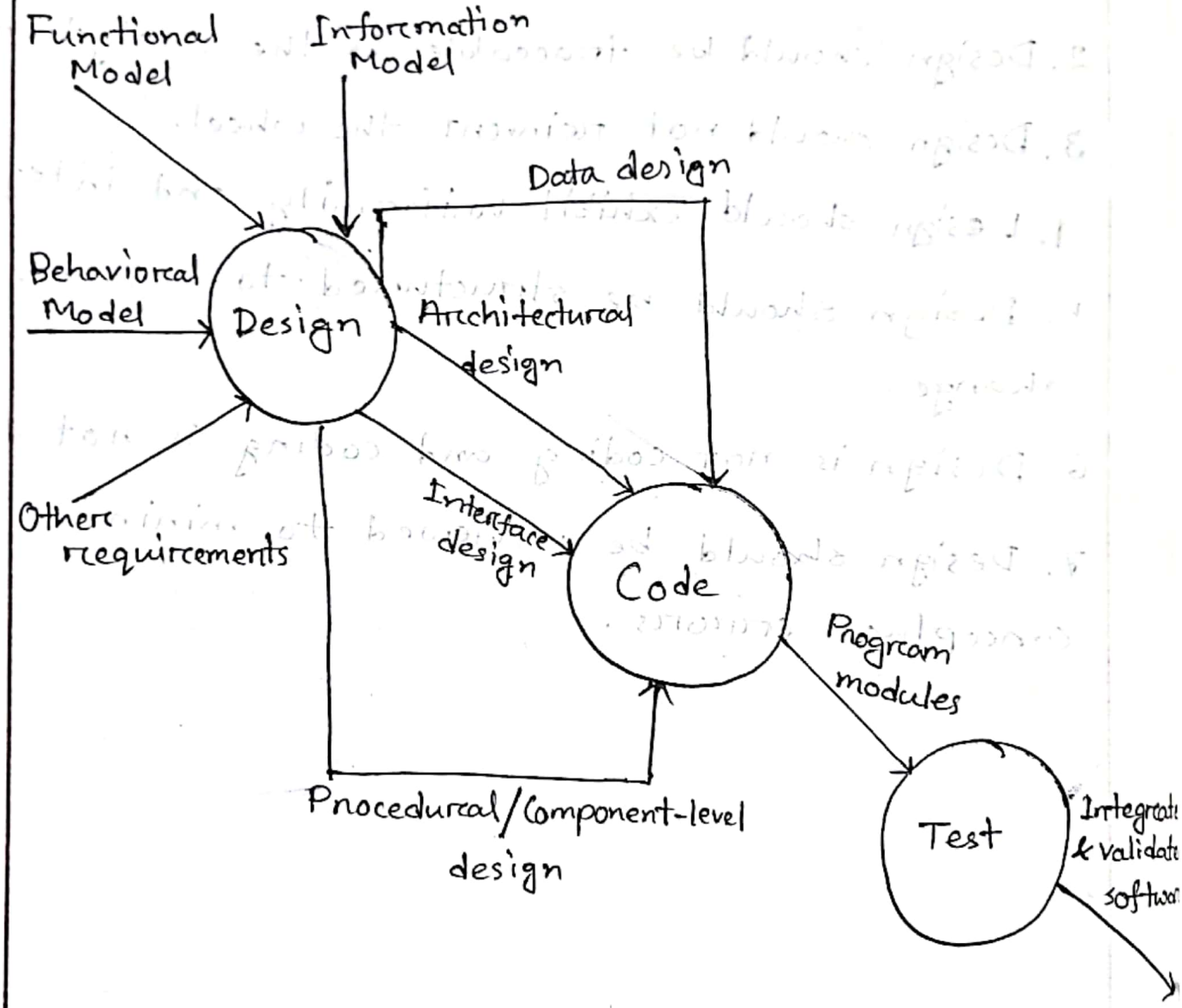
- Lead to data structures that are appropriate for the objects to be implemented.

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

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.

* Data design transforms the information domain model 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.

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 procedu detail without ambiguity.

* Information from process specification, control specification serve as the basis for component desig

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.

- 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 complimentary.

III. Modularity

- Divide software into seperate 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 system. 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.

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.

* 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.

Lecture-6

Chapter-5

S/W Project Planning

(project এর জন্য কত লোক লাগবে তার estimation)

Observations:

- complexity	- size	- Uncertainty
- Relative	- Interdependency	- Hierarchy (info. - Compartmentalization (Fn)

* কতটা complex তা আগে জানা সম্ভব না.
* quantitive estimation দেয়া সম্ভব না first এর দিকে, but implementation এর সময় তা দেয়া সম্ভব.
* size জানা সম্ভব coding এর সময়.
* information কখন দিতে হবে depending on interest.

Resource

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

* Off the shelf components : project এর কিছু component অন্য কোনো organization থেকে নিয়ে আসা.
* Commercial components : অন্য organization থেকে টাকা দিয়ে নিয়ে আসা.
* Full-exp. components : আমাদের project-এ we করা code হুবহুভাবে fully we করতে পারবে. আমাদের team-ই modify করবে.
* Partial-exp. components : আমাদের project কে নতুন কর কম experienced team দিয়ে modify করে we করানো.
* New components : Scratch থেকে project start করা হয়.

first pribility off the staff components যোগাড় বাকি ৩ টের থেকে কম্প খরচ পরবে,

secondly full-exp. components try করা, আগের team বা experienced team ready থাকায় সুবিধা.

3, 4 এর মধ্যে situation depended, যার খরচ কম হবে optimally decide হবে,

কী কী tool/frameworks লাগবে, আর কতদিনের জন্য লাগবে তা আরা, এর জন্য আমাদের কেমন support আছে/ resource আছে,

COCOMO [COnstructive COst MOdel]

	size	Innovation	Deadline	Dev. Env.	kLOC
Organic	S	L	NT	St	2-50
Semi-detached	M	M	M	M	50-300
Embedded	L	G	T	Cx	>300

S - Small
M - Medium
L - Large

Gr - Greeat
T - Tight
NT - Not tight

Cx - Complex
St = Stable
KLOC - Kthousand lines of codes

COCOMO model we করার আগে জানতে হবে কোন classification এ যাবে।

KLOC থেকে direct জানা যায়। KLOC না থাকলে বাকি factors থেকে জানতে হবে।

Stages

- Basic
- Intermediate
- Detail.

Lecture - 7

COCOMO

* Basic

$$\text{Effort} = a_1 (\text{kLOC})^{a_2} \text{ PM} \rightarrow \text{Person-Month}$$

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

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

	a_1	a_2	b
O → 2.4	1.05	0.38
S → 3	1.12	0.35
E → 3.6	1.2	0.32

kLOC দেয়া না থাকলে estimate করতে হবে।

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

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

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

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

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

$$\text{Productivity} = \frac{kLOC}{E}$$

$$= \frac{700}{9341.58}$$

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

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

$E = 2.4 \times (7)^{1.05} = 24.793 \text{ PM} \approx 18.52 \text{ PM}$

$T = 2.5 \times (18.52)^{0.38} = 8.47 \text{ M} = 7.58 \text{ M}$

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

$$\text{Productivity} = \frac{7}{7} \frac{kLOC}{E} = \frac{7}{18.52}$$

$$= \frac{7}{24.793}$$

$$\approx 0.38 \text{ kLOC/PM}$$

$$\approx 0.28 \text{ kLOC/PM}$$

* Intermediate

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

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

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

	a_1	a_2	b
O →	3.2	1.05	0.38
S →	3.0	1.12	0.35
E →	2.8	1.2	0.32

	Very Low	Low	Nominal	High	very High
Product Attributes					
Required SW Reliability	0.75	0.88	1	1.15	1.40
Size of App. DB		0.94	1	1.08	1.16
Capacity of the Product	0.7	0.85	1	1.15	1.3
H/W Attributes					
Runtime Performance Constraints			1	1.11	1.3
Memory Constraints			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 Data~~ Drivers

Cost Drivers

	VL	L	N	H	VH
Personal Attributes					
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	0.95	
Project Attributes					
App & S/W Eng.	1.24	1.1	1	0.91	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 250 KLOC. 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

$$E = 3(250)^{1.12} \times \underbrace{(1 \times 1.06 \times 0.91 \times 1.14 \times 1.08)}_{EAF}$$

$$= 1727.78 \text{ PM}$$

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

$$= 33.97 \text{ M}$$

~~1439 M~~

$$P = \frac{E}{T} = \frac{1727.78}{33.97} = 50.86 \text{ P}$$

$$\simeq 51 \text{ P}.$$

Lecture 9

Software Equation

$$E = \left[\frac{LOC \times B^{0.333}}{P}\right]^3 \times \frac{1}{t^4}$$

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

$$E = 180\, B\, t^3$$

→ in years

B [Special skills factor]

= 0.16 [5-15] KLOC

= 0.39 [>70] KLOC

P [Productivity Parameter]

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

= 10000 [Telecommunication]

= 12000 [Scientific]

= 28000 [Business]

A scientific S/W with approx. 75000 LOC is to be developed. Calculate the time, Effort required for the S/W.

$B = 0.39$

&

$P = 12000$

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

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

$$\simeq 17.89 \text{ M}$$

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

$$\simeq 233 \text{ PM}.$$

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

$B = 0.16$

$P = 20000$

$$t = 8.14 \times \left(\frac{15000}{20000}\right)^{0.43} M$$

$= 19.36 M$

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

$= 120.94 PM$