UNIT-4 : Project Organizations

→ Line-of-business organizations
→ Project organizations
→ evolution of organizations
→ Process automation
→ Project Control & Process instrumentation
→ the seven-core metrics
→ Management indicators
→ quality indicators
→ life-cycle expectations
→ Pragmatic S/w metrics
→ metrics automation

UNIT-4

* Line-of-Business organizations : (LOB)

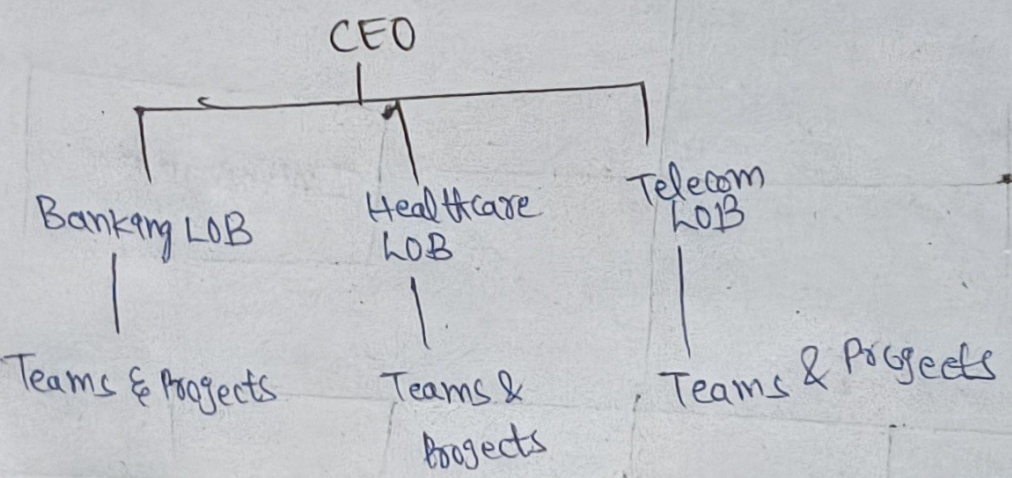
→ A LOB organization means :

- A company is divided into separate business units
- each business unit focuses on specific product/area
- each LOB operates likes a small company inside the main company.

ex: TCS has LOBs like:

- Banking
- Healthcare
- Telecom

→ Each LOB has its own team, budget, goals.



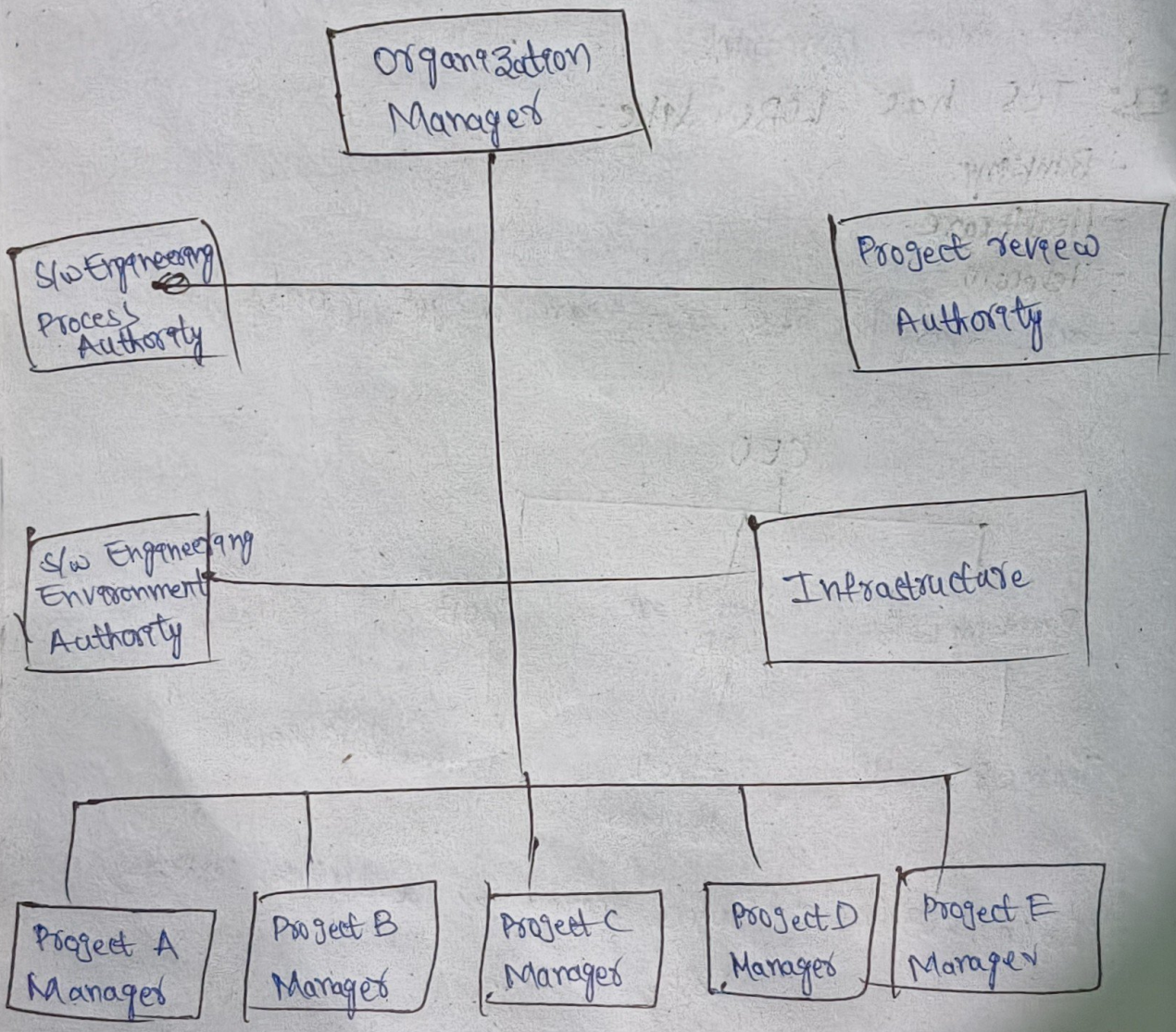
→ Each LOB has : Project managers, Developers, Testers, Sales team.

Features :

- Independent teams
- Flexible
- fast
- has own management

→ LOB organization is a structure
- where a company is divided into multiple business units
- based on product domain like Banking, Telecom.
- Each LOB has its own team, resources

Default roles in LOB



* Project Organization (PO)

→ It means:
- how the Project team is arranged & managed
- who will do what work
- how the team communicates & coordinates

→ It defines Team structure, Roles & responsibilities.

Roles

Project Manager

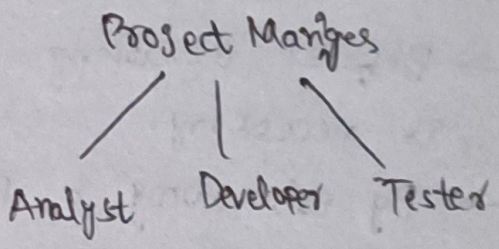
Business Analyst

Developers

Testers

Documentation Engineer

Client



→

(4).

* Evolution of organizations

→ Organizations grow in stages.

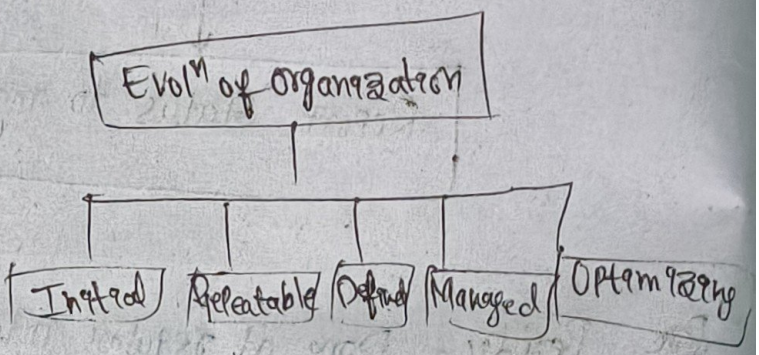
⇒ They move from unstructured → structured → mature → improving.

→ Think of it like:

Baby → Child → Adult → Expert.

Stages

1. Initial
2. Repeatable
3. Defined
4. Managed
5. Optimizing

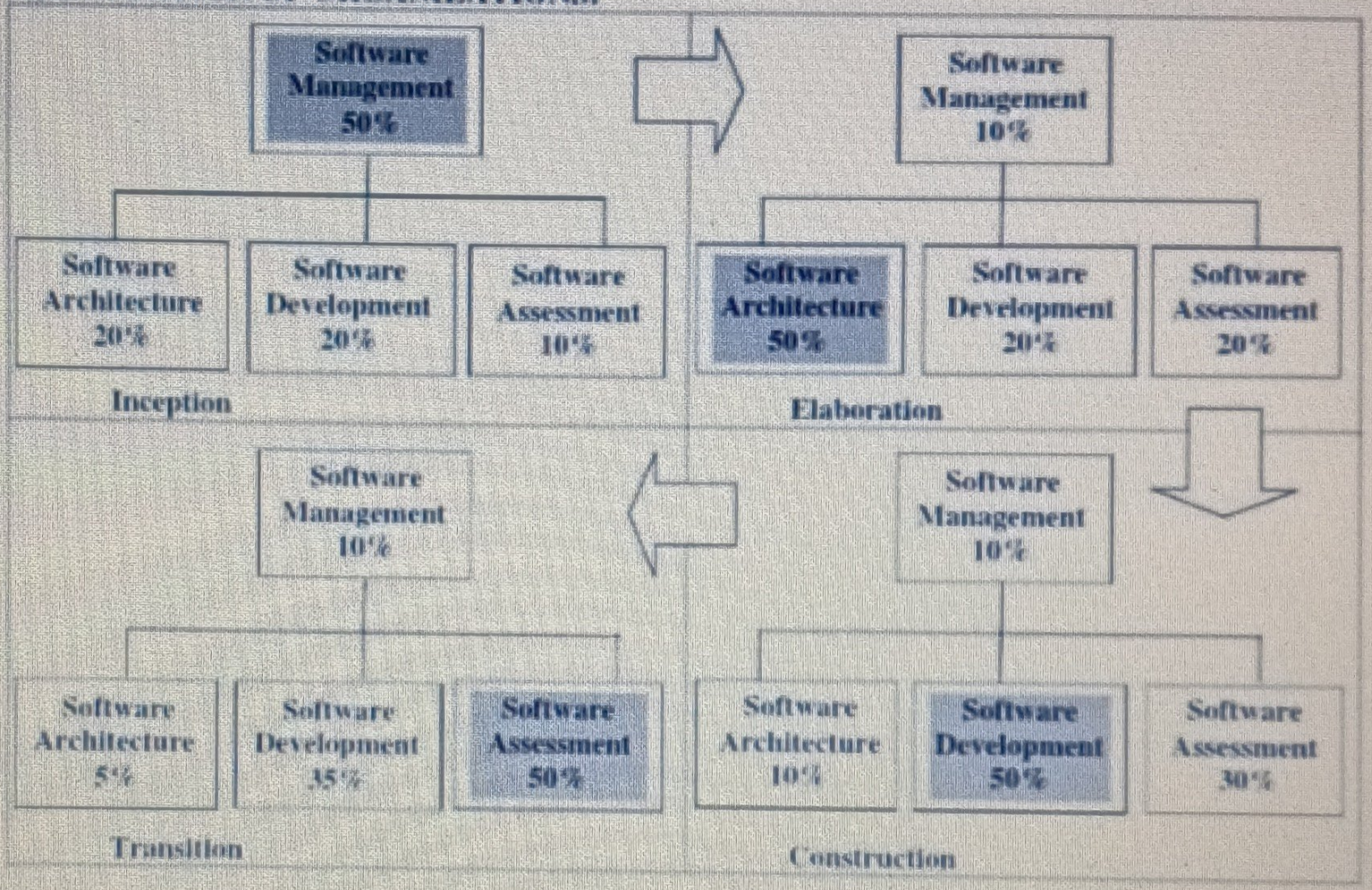


⇒ It refers to the step-by-step improvement of S/w from initial stage to optimizing stage.

⇒ means how an organization improves its S/w processes step-by-step.

→ It show journey from start to end.

3) EVOLUTION OF ORGANIZATIONS:



Inception:
Software management: **50%**
Software Architecture 20%
Software development: 20%
Software Assessment (measurement/evaluation) 10%

Elaboration:
Software management: 10%
Software Architecture: **50%**
Software development: 20%
Software Assessment (measurement/evaluation) 20%

Construction:
Software management: 10%
Software Architecture 10%
Software development: **50%**
Software Assessment (measurement/evaluation) 30%

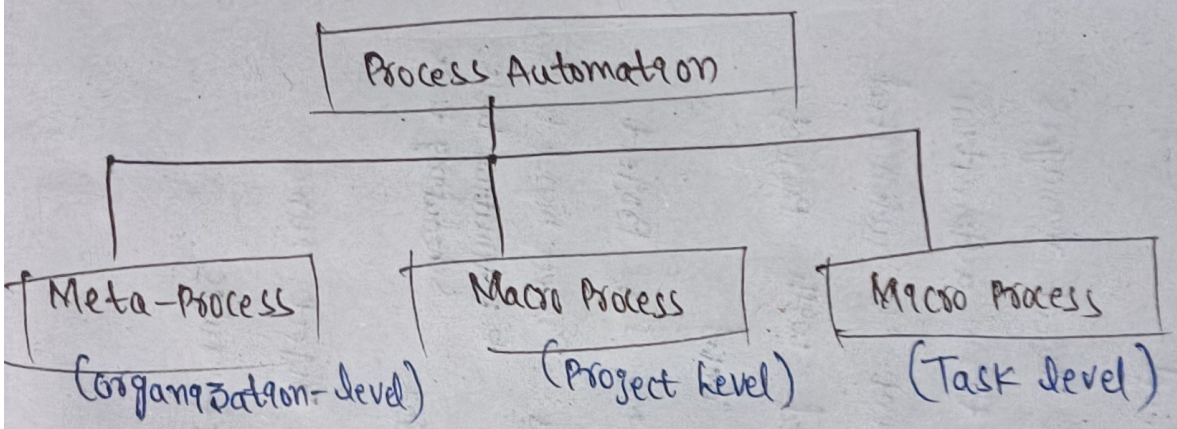
Transition:
Software management: 10%
Software Architecture: 5%
Software development: 35%
Software Assessment (measurement/evaluation): **50%**

* Process Automation

→ Automation helps organizations to perform tasks faster, accurate
- by using tools, scripts & systems.

→ Automation happens at ③ levels:

① Meta Process Level
② Macro Process Level
③ Micro Process Level.



Meta-Process Level

1). Highest level
2). used by organization
3). Long-term automation
4). Infrastructure is automated
5). Associated with organization
6). organization level
7). Automation for entire organization
8). It includes: Policies, Procedures, Process definition tools.

9). eg: how an organization runs
- Company rules

Macro Process Level

1). Middle level
2). used by Project managers
3). Medium-term automation
4). Environment is automated
5). Associated with Project
6). Project level
7). Automation for Project
8). It includes: Project mgmt tools, & Estimation tools, Scheduling tools

ex: Jira, Git

9). eg: how a Project runs
- Project rules

Micro Process Level

1). Lowest level.
2). used by devlopers/team.
3). Short-term automation
4). Tools are automated.
5). Associated with team ~~member~~
6). Developer level
7). Automation for tasks.
8). It includes: Testing tools, Build tools, Debugging tools.

ex: Compiler, debugger

9). eg: how a small task runs
- Task rules

1. Project Control

What it is:

- Project control is the **process of monitoring a project** to make sure it **stays on track** with schedule, cost, and quality.

Meaning:

- Ensuring the project is being **executed as planned**.
- If there's a **deviation**, taking **corrective action** immediately.

Details:

- Purpose:
  - Detect **problems early**.
  - Ensure project meets **time, cost, and quality goals**.
  - Support **decision-making** for adjustments.
- How it's done:
  1. **Set standards & plan** – schedule, budget, quality metrics.
  2. **Measure actual performance** – track progress, cost, defects.
  3. **Compare with plan** – see deviations.

- **Purpose:**
  - Detect **problems early**.
  - Ensure project meets **time, cost, and quality goals**.
  - Support **decision-making** for adjustments.
- **How it's done:**
  1. **Set standards & plan** – schedule, budget, quality metrics.
  2. **Measure actual performance** – track progress, cost, defects.
  3. **Compare with plan** – see deviations.
  4. **Take corrective action** – reallocate resources, revise schedule, fix issues.

Small Example:

- Planned: Complete 10 modules in 5 weeks.
- Actual: 7 modules done in 5 weeks → Project is **behind schedule**, action needed.

Real-World Example:

- In a software company, **weekly project meetings** check progress against milestones. If coding is delayed, resources may be reassigned.

↓

What it is:

- Process instrumentation is **using metrics, tools, and techniques** to **measure and control the software process.**

Meaning:

- "Instrumenting" = **measuring and monitoring.**
- Helps **know how the process is performing** so you can **control it effectively.**

Details:

- **Purpose:**
  - Ensure the software process is **efficient and predictable.**
  - Identify **bottlenecks or problem areas.**
- **Common Metrics Used:**
  - **Schedule metrics:** % tasks completed, milestone achievement.
  - **Cost metrics:** Actual vs. planned cost.
  - **Quality metrics:** Defect density, test coverage, customer satisfaction.

- **Tools:** Charts, dashboards, automated trackers, bug tracking tools.

Small Example:

- Track **defects per 1000 lines of code** weekly to monitor code quality.

Real-World Example:

- In Agile, **burn-down charts** track remaining work vs. time to ensure sprint completion.

✅ Summary to remember:

Term	Meaning	Purpose	Example
Project Control	Monitoring & correcting project execution	Stay on schedule, budget, quality	Weekly project review
Process Instrumentation	Using metrics/tools to monitor software process	Improve predictability & quality	Burn-down chart, defect density

- **Lifecycle Expectations = what the project should achieve at each phase** of development.

---

2. Phases & Expectations (Easy Table)

Phase	Expectation (in simple words)
Requirements	Know exactly what the user wants
Design	Plan how the software will work
Implementation	Write correct and clean code
Testing	Find and fix errors
Deployment	Make software work for users
Maintenance	Keep it running, fix problems, update

3. Easy Example

- Making a **shopping app:**
  - Requirements → List of features
  - Design → App layout & structure
  - Implementation → Code the app
  - Testing → Fix bugs
  - Deployment → App live for users
  - Maintenance → Update features & fix issues

Tip to Remember

- R-D-I-T-D-M → Requirements, Design, Implementation, Testing, Deployment, Maintenance
- Think: "Real Developers Implement Things Daily, Mindfully" 😄

1. What are Pragmatic Software Metrics?

- **Pragmatic Software Metrics** are **practical measurements** used to **monitor and control software projects.**
- They are **easy to collect, understand, and act upon** by the project team.

---

2. Meaning

- "Pragmatic" → **practical, useful in real-world situations.**
- "Software Metrics" → **numbers or measurements** that indicate project progress, quality, or performance.
- Combined → Metrics that **help managers make real decisions** without being too complicated.

3. Details / Explanation

- Purpose:
  - Track **project progress** and **product quality.**
  - Identify **problems early.**
  - Improve **processes and planning.**

- **Characteristics of Pragmatic Metrics:**
  1. **Simple to understand** – Team can use without extra training.
  2. **Easy to collect** – Doesn't require complex tools.
  3. **Actionable** – Leads to decisions or corrective action.
  4. **Relevant** – Directly tied to project goals.
- **Examples of Pragmatic Metrics:**
  - **Schedule metrics:** % of tasks completed, milestone achievement.
  - **Cost metrics:** Actual vs. planned expenditure.
  - **Quality metrics:** Number of defects per module, test coverage.
  - **Productivity metrics:** Lines of code per day, features delivered per week.

1. What is Metrics Automation?

- **Metrics Automation** is the **use of tools and software** to automatically **collect, calculate, and repor**
  software metrics.
- It reduces **manual effort** and **improves accuracy and speed** in project monitoring.

2. Meaning

- "Metrics" → Measurements of project performance, quality, cost, etc.
- "Automation" → Doing it automatically using software/tools.
- Combined → **Automatically tracking project health using metrics.**

3. Details / Explanation

- Purpose:
  1. Save **time and effort** in collecting metrics.
  2. Ensure **accuracy** (less human error).
  3. Provide **real-time monitoring** of project status

3. Details / Explanation

- **Purpose:**
  1. Save **time and effort** in collecting metrics.
  2. Ensure **accuracy** (less human error).
  3. Provide **real-time monitoring** of project status.
  4. Support **quick decision-making** for corrective actions.
- **Typical Metrics Automated:**
  - **Schedule metrics:** % tasks completed, milestone tracking.
  - **Cost metrics:** Actual vs planned cost.
  - **Quality metrics:** Defect density, test coverage, bug trends.
  - **Productivity metrics:** Features delivered, lines of code per time.
- **Tools for Metrics Automation:**
  - **Jira / Trello / Azure DevOps:** Track task completion.
  - **SonarQube:** Code quality, bug density.
  - **CI/CD Tools (Jenkins, GitLab):** Build and test metrics.
  - **Excel / Power BI dashboards:** Auto-generate charts.

4. Small Example

- A team uses **Jira** to track user stories.
- The tool automatically shows:
  - How many stories are done this week.
  - How many are delayed.
  - Defects reported.
- No manual calculation is needed → **project status is visible in real-time.**