innovate achieve lead

BITS Pilani
Pilani Campus

SE ZG501
Software Quality Assurance and Testing
session No. 12

Test Process Improvement (TPI) Next

- The Test Process Improvement Next model, also called TPI Next, is the next generation of its predecessor TPI (Test Process Improvement) model.
- TPI Next aims to strongly establish its predecessor's strengths while keeping in focus the business goals of the organization.
- Since TPI Next aims at the business-related key areas, it can be referred to for activities beyond testing and is a generic model to implement.

TPI Next model describes four maturity levels:

Maturity Level 1 – Initial

At this level, no formal process is followed.The testing process is chaotic, unstructured, and highly inconsistent .There are no rules or standard practices, and testing is usually adhoc. The organization operates with whatever approach is currently in place, often leading to high defect rates and inefficiency..

Maturity Level 2 – Controlled

A basic structure is introduced to make the process more organized and manageable. Stakeholders begin to participate actively in streamlining testing activities. This level helps in reducing bugs but is still far from optimal. The process is controlled but not yet optimized, and many bugs may still exist.

Maturity Level 3 – Efficient

At this stage, the focus is on achieving **efficiency in existing testing processes**.

- Redundant or overlapping testing steps are identified and **eliminated**.
- Efforts are made to **streamline** the workflow and **reduce time wastage**.
- Processes are **integrated into a unified and structured testing cycle**.
- Goal: Minimize inefficiencies and ensure testing is lean and well-coordinated.

Maturity Level 4 – Optimizing

This is the **highest level of maturity** in the TPI Next model.

- A team of coordinators or experts **continuously evaluates** the testing process.
- Insights from **past projects (bugs, delays, challenges)** are used to make improvements.
- The aim is **continuous optimization** for future projects.
- Testing is now **adaptive, preventive, and driven by feedback and learning**.

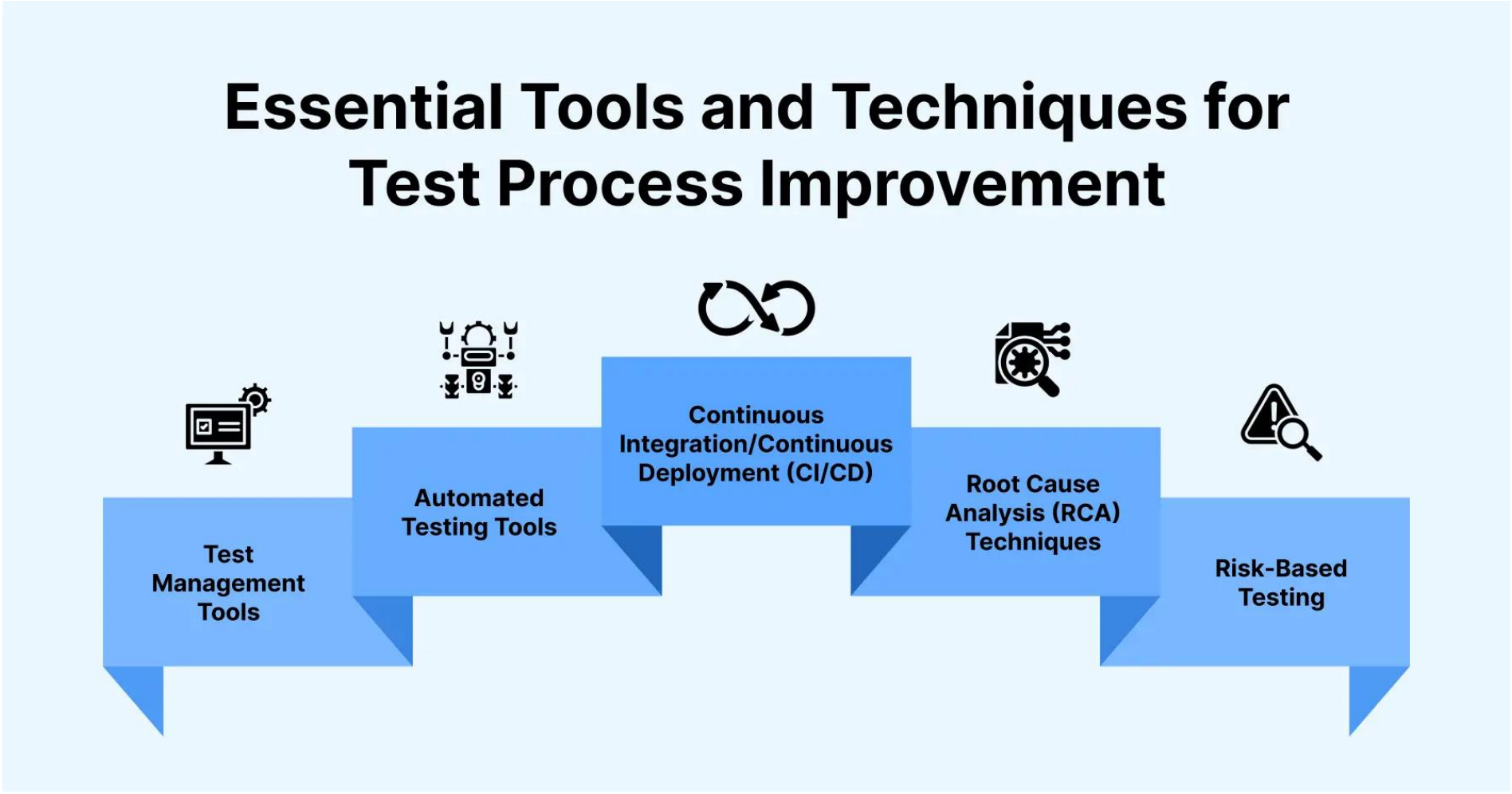
Systematic Test and Evaluation Process (STEP)

The STEP model focuses on testing software requirements systematically from the beginning of the Software Development Life Cycle (SDLC). It ensures better quality by evaluating results early and optimizing the process before coding begins.

Characteristics of STEP

- STEP focuses on requirement-driven testing
- STEP aims at performing testing first and then starting coding.
- STEP helps in early defect detection due to the “testing first” approach.

Tools and Techniques for Test Process Improvement



Test Management Tools :

- To successfully improve the test process, test management must be effective.
- Teams can effectively plan, carry out, and monitor testing operations using test management solutions like Jira, TestRail, and HP ALM (Application Lifecycle Management).

Automated Testing Tools:

- Teams may automate time-consuming and repetitive test cases using tools like Selenium, Appium, and JUnit, freeing up testers to concentrate on more important scenarios.
- Automation broadens the scope of tests, quickens the testing process, and improves the accuracy of test results. Additionally, it makes early bug discovery easier, which lowers the expense and work needed to correct flaws.

Continuous Integration/Continuous Deployment (CI/CD):

- Implementing CI/CD pipelines guarantees that software be released more quickly and frequently without sacrificing quality.
- The processes for building, integrating, and deploying software are automated by tools like Jenkins, GitLab CI/CD, and CircleCI.
- Organizations may do continuous testing, identify issues early, and establish a smooth and reliable deployment process by integrating automated tests into the CI/CD pipeline.

Root Cause Analysis (RCA) Techniques:

RCA is a method used to **find out the underlying reason** for defects or problems in the testing process.

It helps prevent the same issues from happening again by **fixing the real cause**, not just the symptoms.

Example: If frequent bugs are found in one module, RCA might reveal that unclear requirements are the actual issue.

Risk-Based Testing

Risk-Based Testing prioritizes testing efforts based on **potential risk and impact**.

Features or components that are **more likely to fail** or cause **serious problems** are tested first.

Example: In a banking app, login and payment systems are tested more thoroughly than the user profile section.

CMMi

- The Capability Maturity Model Integration (CMMI) is a model that aids in identifying the strengths and weaknesses of an organization's current processes and shows the way to improvement.
- CMMI's primary goal is to create high-quality software.

- CMMI is a technology offered by SEI that assists businesses in standardizing **software development**, **testing**, and **deployment** to improve the product's quality.
- The implementation of standards that will assist raise the quality of the software products is supported by CMMI.
- According to CMMI, a complete model consisting of five "Maturity Levels" that is essential to creating excellent software.
- CMMI is a more advanced model of its CMM predecessor.

What is CMM?

CMM: Capability Maturity Model

Developed by the **Software Engineering Institute of the Carnegie Mellon University**

Framework that **describes the key elements of an effective software process**.

What is CMM?

Describes an **evolutionary improvement path** for software organizations from an ad hoc, immature process to a mature, disciplined one.

Provides guidance on how to gain control of processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence.

Process Maturity Concepts

Software Process

- Set of **activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, user manuals)**

Software Process Capability

- Describes the range of expected results **that can be achieved by following a software process**
- Means of predicting the most likely outcomes **to be expected from the next software project the organization undertakes**

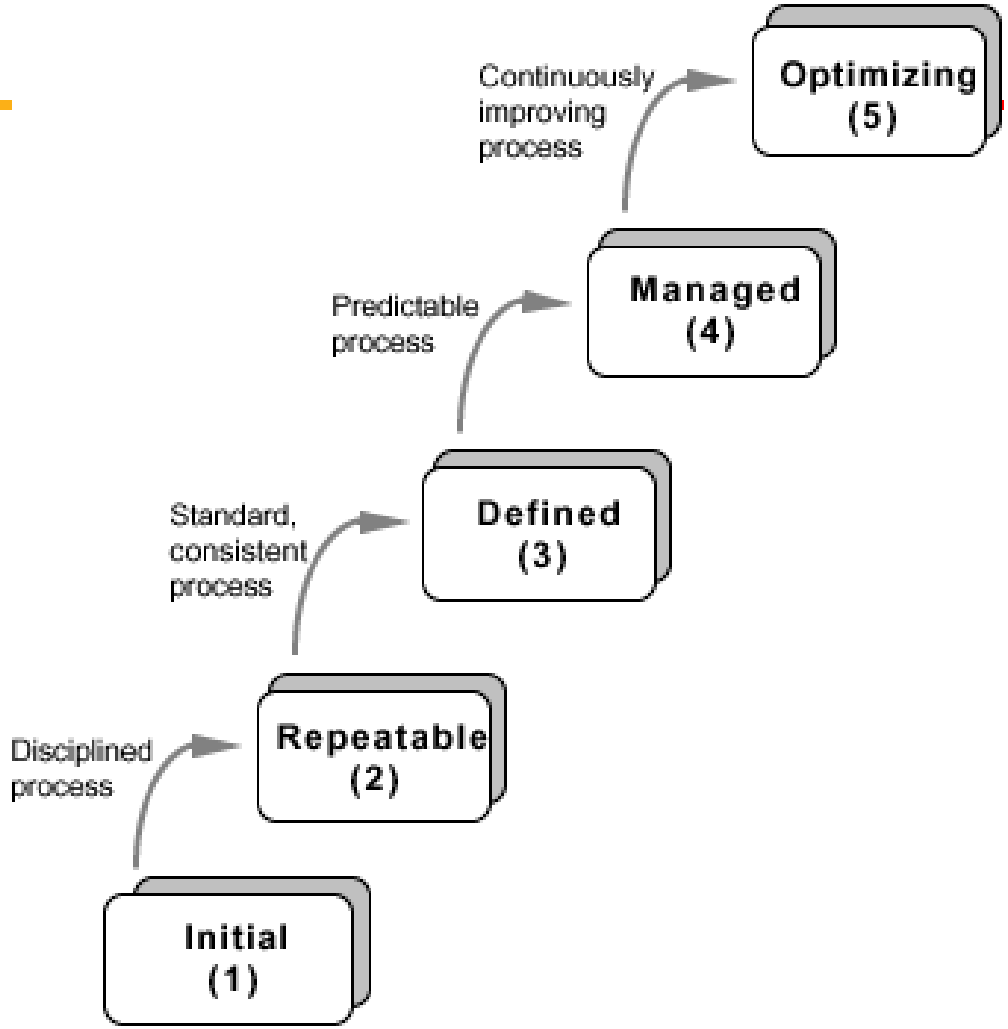
Process Maturity Concepts

- **Software Process Performance**
  - Actual results achieved **by following a software process**
- **Software Process Maturity**
  - Refers to how well a process is **clearly defined, managed, measured, and controlled** to ensure its effectiveness.
  - It reflects the organization's ability to **grow and improve** and shows how **consistently and thoroughly** the process is applied across all projects.

What are the CMM Levels?
(The five levels of software process maturity)

Maturity level indicates level of process capability:

- Initial
- Repeatable
- Defined
- Managed
- Optimizing


Optimizing
(5)
Continuously
improving
process
Managed
(4)
Predictable
process
Defined
(3)
Standard,
consistent
process
Repeatable
(2)
Disciplined
process
Initial
(1)

Level 1: Initial

- The software process is ad hoc and sometimes even chaotic. There are very few defined processes, and success depends mostly on individual effort.
- At this level, the team often struggles to make commitments in an orderly manner.
- Products are usually completed over budget and behind schedule. There are large variations in cost, quality, schedule, and functionality. The capability resides in individuals rather than in the organization as a whole.

Level 2: Repeatable

Basic process management practices are established to **monitor cost, schedule, and functionality**. A consistent process discipline allows teams to repeat successful outcomes from previous similar projects.

Realistic project commitments are made based on past experiences. Software standards are defined and followed. Processes may vary between projects but are carried out in a disciplined manner. Successes from earlier projects can be reliably repeated.

Level 3: Defined

- The software process for both management and engineering activities is **documented, standardized, and integrated** into a defined software process across the organization.
- All projects follow an **approved and tailored version** of this standard process, ensuring consistency in software development and maintenance practices.

Level 4: Managed

- At this level, detailed measurements of the software process and product quality are collected. Both are quantitatively understood and controlled.
- Process variations are kept within acceptable limits.
- Corrective actions are taken when performance deviates from expected bounds.
- The process is quantifiable, predictable, and can be used to forecast trends in process and product quality.

Level 5: Optimizing

- Continuous improvement is driven by quantitative feedback from the process and by piloting innovative ideas and technologies.
- The focus is on preventing defects through causal analysis and proactive measures.
- Process effectiveness data is used for cost-benefit analysis of new technologies and to support proposed process changes, ensuring ongoing optimization and innovation.

Break-time Brain Teaser:

An IT company has reached a high level of process maturity. It uses defect trend data to identify recurring issues, applies root cause analysis to prevent future defects, and runs pilot projects to evaluate the benefits of new development tools before adopting them company-wide.

Which of the following practices indicate the organization is operating at CMMI Level 5 – Optimizing?
(Select all that apply)

a) Using root cause analysis to prevent defects
b) Conducting pilot studies for new technologies
c) Following basic project management practices
d) Applying data-driven feedback for continuous improvement
e) Relying only on historical practices without change

Break-time Brain Teaser:

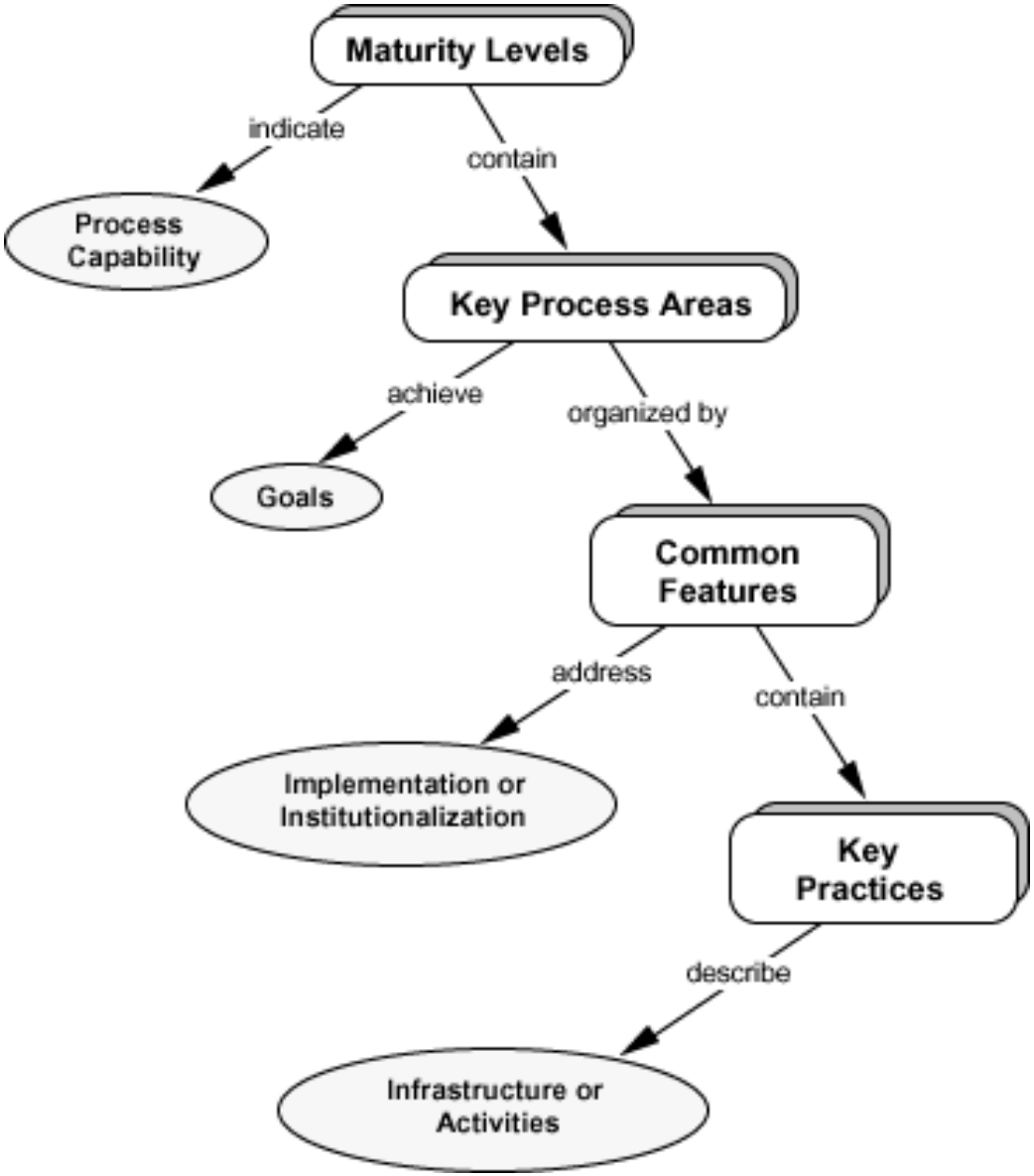
innovate achieve lead

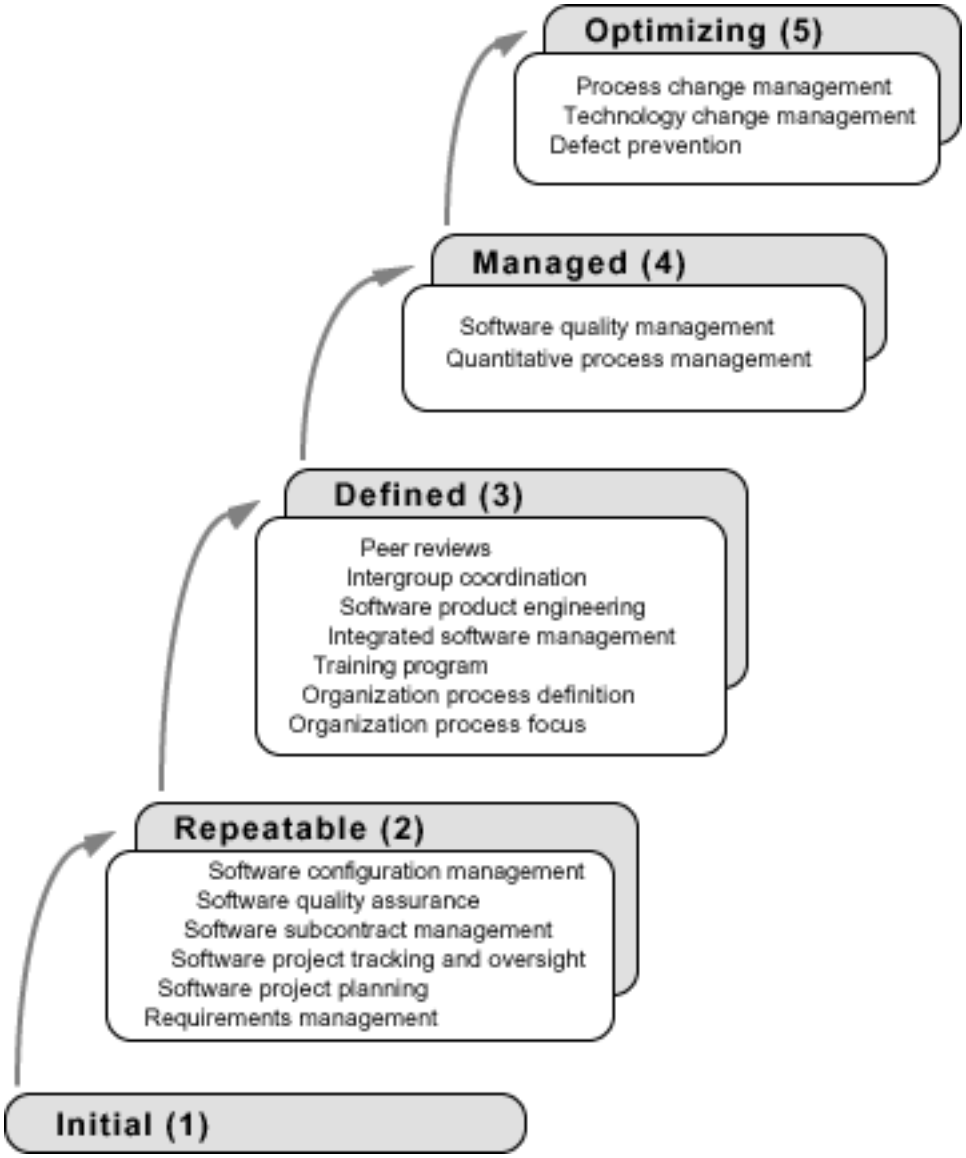
Answer:

a) Using root cause analysis to prevent defects
b) Conducting pilot studies for new technologies
d) Applying data-driven feedback for continuous improvement

Internal Structure to Maturity Levels

- Except for level 1, each level is decomposed into key process areas (KPA)
- Each KPA identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing software capability.
  - commitment
  - ability
  - activity
  - measurement
  - verification


Maturity Levels
indicate
contain
Process Capability
Key Process Areas
achieve
organized by
Goals
Common Features
address
contain
Implementation or Institutionalization
Key Practices
describe
Infrastructure or Activities



The Key Process Areas by Maturity Level

Level 2 KPAs

Requirements Management

- Establish common understanding of customer requirements between the customer and the software project
- Requirements is basis for planning and managing the software project
- Not working backwards from a given release date!

Software Project Planning

- Establish reasonable plans for performing the software engineering activities and for managing the software project.

Level 2 KPAs

Software Project Tracking and Oversight

- Establish adequate visibility into actual progress
- Take effective actions when project's performance deviates significantly from planned

Software Subcontract Management

- Manage projects outsourced to subcontractors

Software Quality Assurance

- Ensure visibility into the process and the quality of work products.

Level 2 KPAs

Software Configuration Management

- Track and control changes to software and documents.
- Maintain the integrity of all work products.
- Use product baselines as approved reference versions.
- Allow changes only through authorized baseline control.

Level 3 KPAs

1. **Organization Process Focus**
   - Assign teams to improve and manage software processes.
   - Goal: Strengthen the organization's overall development process.
2. **Organization Process Definition**
   - Create and maintain a standard set of process guidelines.

- **Provides a stable base for:**
  - Consistent use across projects.
  - Collecting useful data for improving processes.

Level 3 KPAs

Training Program:

- Ensure team members have the right skills and knowledge.
- Training is based on project needs and managed by the organization.

Integrated Software Management

- Combine engineering and management activities.
- Use standard processes, customized for specific project and business needs.

Level 3 KPAs

Software Product Engineering

•All technical work (like in the Software Development Life Cycle) is clearly planned.

•Work results are accurate and consistent.

Intergroup Coordination

•Software teams work well and actively with other departments.

Peer Reviews

•Helps find and fix problems early.

•Improves understanding of the product.

•Done using reviews like inspections and walkthrough

Level 4 KPAs

innovate achieve lead

Quantitative Process Management

- Use numbers and data to measure how well the process is working.
- Compare actual results with expected results.
- Find and fix problems that cause unexpected changes in the process.
- **Software Quality Management**
- Use data to understand and improve software quality.
- Focus on both:
  - the **product** (the software itself)
  - the **process** (how the software is developed)

Level 5 KPAs

1. **Process Change Management**
    1. Keep improving processes to:
        1. make better quality products
        2. work faster
        3. increase productivity
2. **Technology Change Management**
    1. Find and use helpful new tools, methods, or processes to improve work.
3. **Defect Prevention**
    1. Understand why defects happen and stop them from happening again.

What are the benefits ?

- Helps forge a shared vision of what software process improvement means for the organization
- Defines set of priorities for addressing software problems
- Supports measurement of process by providing framework for performing reliable and consistent appraisals
- Provides framework for consistency of processes and product

Why measure software and software process?

Obtain data that helps us to better control:

- Schedule
- Cost
- Quality of software products

Consistent measurement provide data for:

- Define clear goals, requirements, and success criteria
- Track progress and detect problems early
- Make smart decisions about resource allocation
- Accurately estimate time, cost, and quality

Measurements

1. **Historical** – Data from past projects (used for comparison and learning).
2. **Plan** – What we expect to happen (like timelines and costs).
3. **Actual** – What really happened during the project.
4. **Projections** – Forecasts for future progress based on current trends.

SEI Core Measures

Unit of Measure	Characteristics Addressed
Physical source lines of code Logical source lines of code	Size, reuse, rework
Staff hours	Effort, cost, resource allocations
Calendar dates for process milestones Calendar dates for deliverables	Schedule, progress
Problems and defects	Quality, improvement trends, rework, readiness for delivery

Examples of measurements for size of work products

- Estimated number of requirements
- Actual number of requirements
- Estimated source lines of code (SLOC)
- Actual SLOC
- Estimated number of test cases
- Actual number of test cases

Example of measurements of effort

- Estimated man-hours to design/code a given module
- Actual man-hours expended for designing /coding the module
- Estimated number of hours to run builds for a given release
- Actual number of hours spent running builds for the release

Examples of measurements of quality of the work product

- Number of issues raised at requirements inspection
- Number of requirements issues open
- Number of requirements issues closed
- Number of issues raised during code inspection
- Number of defects opened during unit testing

Examples of measurements of quality of the work product

- Number of defects opened during system testing
- Number of defects opened during UAT
- Number of defects still open
- Number of defects closed
- Defect age

Examples of measurements of quality of the work product

- Total number of build failures
- Total number of defects fixed for a given
- release
- Total number of defects verified and accepted
- Total number of defects verified and rejected