DIGITAL NOTES ON SOFTWARE PROCESS AND PROJECT MANAGEMENT [R18]

B.TECH IV YEAR - I SEM

(2022-23)

BVRIT HYDERABAD College of Engineering for Women

DEPARTMENT OF INFORMATION TECHNOLOGY

CS725PE: SOFTWARE PROCESS & PROJECT MANAGEMENT
(Professional Elective - V)

IV Year B.Tech. IT I -Sem

L	T	P	C
3	0	0	3

Course Objectives:

- To acquire knowledge on software process management
- To acquire managerial skills for software project development
- To understand software economics

Course Outcomes:

- Gain knowledge of software economics, phases in the life cycle of software development, project organization, project control and process instrumentation
- Analyze the major and minor milestones, artifacts and metrics from management and technical perspective
- Design and develop software product using conventional and modern principles of software project management

UNIT - I

Software Process Maturity

Software maturity Framework, Principles of Software Process Change, Software Process Assessment, The Initial Process, The Repeatable Process, The Defined Process, The Managed Process, The Optimizing Process.

Process Reference Models

Capability Maturity Model (CMM), CMMI, PCMM, PSP, TSP).

UNIT - II

Software Project Management Renaissance

Conventional Software Management, Evolution of Software Economics, Improving Software Economics, The old way and the new way.

Life-Cycle Phases and Process artifacts

Engineering and Production stages, inception phase, elaboration phase, construction phase, transition phase, artifact sets, management artifacts, engineering artifacts and pragmatic artifacts, model-based software architectures.

UNIT - III

Workflows and Checkpoints of process

Software process workflows, Iteration workflows, Major milestones, minor milestones, periodic status assessments.

Process Planning

Work breakdown structures, Planning guidelines, cost and schedule estimating process, iteration planning process, Pragmatic planning.

UNIT - IV

Project Organizations
Line-of- business organizations, project organizations, evolution of organizations, process automation. Project Control and process instrumentation
The seven-core metrics, management indicators, quality indicators, life-cycle expectations,

Pragmatic software metrics, metrics automation.

UNIT - V

CCPDS-R Case Study and Future Software Project Management Practices
Modern Project Profiles, Next-Generation software Economics, Modern Process Transitions.

TEXT BOOKS:

1. Managing the Software Process, Watts S. Humphrey, Pearson Education
2. Software Project Management, Walker Royce, Pearson Education

REFERENCE BOOKS:

1. An Introduction to the Team Software Process, Watts S. Humphrey, Pearson Education, 2000
2. Process Improvement essentials, James R. Persse, O'Reilly, 2006
3. Software Project Management, Bob Hughes & Mike Cotterell, fourth edition, TMH, 2006
4. Applied Software Project Management, Andrew Stellman & Jennifer Greene, O'Reilly, 2006.
5. Head First PMP, Jennifer Greene & Andrew Stellman, O'Reilly, 2007
6. Software Engineering Project Management, Richard H. Thayer & Edward Yourdon, 2nd edition, Wiley India, 2004.
7. Agile Project Management, Jim Highsmith, Pearson education, 2004.

UNIT I

Software Process Maturity

Software maturity Framework**,** Principles of Software Process Change, Software Process Assessment, The Initial Process, The Repeatable Process, The Defined Process, The Managed Process, The Optimizing Process.

Process Reference Models

Capability Maturity Model (CMM)**,** CMMI, PCMM, PSP, TSP.

Part-1

Software Process:

A Software Process is a set of related activities that leads to the production of software product.

Four activities that are fundamental to software engineering:

- **Software Specification** : The functionality of the software and constraints on its operation must be defined.
- **Software Design and Implementation** : The software to meet the specification must be produced.
- **Software Validation:** The software must be validated to ensure that it does what customer wants.
- **Software Evolution:** The software must evolve to meet changing customer needs.

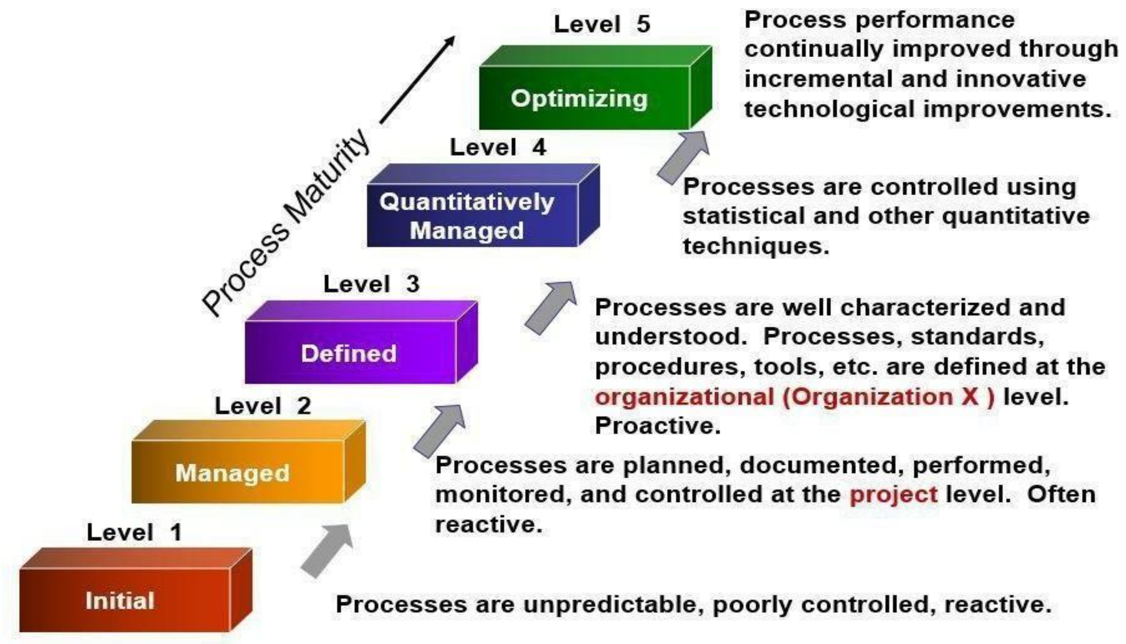
Software Maturity Framework

The CMM focuses on the capability of software organizations to produce high- quality products consistently and predictably. Software process capability is the inherent ability of a software process to produce planned results.

- **Process** A sequence of steps performed for a given purpose. The process integrates people, tools, and procedures.
- **Software Process** A set of activities, methods, practices, and transformations that people employ to develop and maintain software and the associated products (documents, etc.)
- **Software Process Capability** describes the range of expected results that can be achieved by following a software process.
- **Software Process Performance** the actual results achieved by following a software process.
- **Software Process Maturity** the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. As a software organization matures, it needs an infrastructure and culture to support its methods, practices, and procedures so that they endure after those who originally defined them have gone.
- **Institutionalization** is the building of infrastructure and culture to support methods, practices, and procedures so that they are the ongoing way of doing business.

Software Process Maturity Framework

Five Maturity Levels:


Process Maturity
Level 5
Optimizing
Process performance continually improved through incremental and innovative technological improvements.
Level 4
Quantitatively Managed
Processes are controlled using statistical and other quantitative techniques.
Level 3
Defined
Processes are well characterized and understood. Processes, standards, procedures, tools, etc. are defined at the organizational (Organization X) level. Proactive.
Level 2
Managed
Processes are planned, documented, performed, monitored, and controlled at the project level. Often reactive.
Level 1
Initial
Processes are unpredictable, poorly controlled, reactive.

- **Initial:** The software process is characterized by ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
- **Repeatable:** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
- **Defined:** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization’s standard software process for developing and maintaining software.
- **Managed:** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitative understood and controlled.
- **Optimizing:** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

<u>Principles of Software Process Change</u>

Basic Principles

- Automation of a poorly defined process will produce automation of poorly defined results
- Improvement should be made in some steps

- Educate/Train, Educate/Train, , Educate/Train

Principles of Software Process Change

Software process management has 2 key areas

People

Design methods

People: A good mix of talent is required. The best people are always in short supply. You probably have about the best team you can get right now. With proper leadership, education, training and support, most people can do better work than they are currently doing.
Design: When domain knowledge is combined with the ability to produce a good design, a quality product will result.

Six Basic Principles of Software Process Change

1. **Major changes to the software process must start at the top**
   - Major changes requires leadership. Managers must provide good leadership, even though they may not do the work; they must set priorities: furnish resources and provide continuing support.
2. **Ultimately, everyone must be involved.**
   - With an immature software process, software professionals are forced to improvise solutions. in a mature process, these individual actions are more structured, efficient and reinforcing. People are the most important aspect. It's necessary to focus on repairing the process and not the people.
3. **Effective changes require the team to have common goals and knowledge of the current process.**
   - An effective change program requires a reasonable understanding of the current status. An assessment is an effective way to gain this understanding. Software professionals generally need most help in controlling requirements., coordinating changes, making plans. managing interdependencies and coping with system design issues.

4. **Change is continuous**
   - One of the most difficult things for a management team to recognize is that human interactive processes are never static. Both problems and people are in constant flux, and this fluidity all for periodic adjustment of tasks and relationships. In dealing with these dynamics, 3 points are important.
   - Relative changes generally make things worse
   -Every defect is an improvement opportunity
   -Crisis prevention is more important than crisis recovery
5. **Software process changes will not be retained without conscious effort and periodic reinforcements**
   - Precise and accurate work is hard. Its rarely sustained for long without reinforcement. Human adoption of new process methods involves 4 stages
     - Installation, Practice , Proficiency , Naturalness
6. **Software process improvement requires investment**
   - While the need for dedicating resources to improvement seems self- evident it's surprising how often managers rely on exhorting their people to try harder
   - To improve the aft rare process someone must work on it
   - Unplanned process improvement is wishful thinking.

Software Process Assessment:

A software process assessment is a disciplined examination of the software processes used by an organization, based on a process model. The assessment includes the identification and characterization of current practices, identifying areas of strengths and weaknesses, and the ability of current practices to control or avoid significant causes of poor (software) quality, cost, and schedule.

A software assessment (or audit) can be of three types.

A **self-assessment (first-party assessment)** is performed internally by an organization's own personnel.

A **second-party assessment** is performed by an external assessment team or the organization is assessed by a customer.

A **third-party assessment** is performed by an external party or (e.g., a supplier being assessed by a third party to verify its ability to enter contracts with a customer).

Software process assessments are performed in an open and collaborative environment. They are for the use of the organization to improve its software processes, and the results are confidential to the organization. The organization being assessed must have members on the assessment team.

Software Process Maturity Assessment

The scope of a software process assessment can cover all the processes in the organization, a selected subset of the software processes, or a specific project. Most of the standard-based process assessment approaches are invariably based on the concept of process maturity.

When the assessment target is the organization, the results of a process assessment may differ, even on successive applications of the same method.

There are two reasons for the different results. They are,

The organization being investigated must be determined. For a large company, several definitions of organization are possible and therefore the actual scope of appraisal may differ in successive assessments.

Even in what appears to be the same organization, the sample of projects selected to represent the organization may affect the scope and outcome.

When the target unit of assessment is at the project level, the assessment should include all meaningful factors that contribute to the success or failure of the project. It should not be limited by established dimensions of a given process maturity model. Here the degree of implementation and their effectiveness as substantiated by project data are assessed.

Process maturity becomes relevant when an organization intends to embark on an overall long- term improvement strategy. Software project assessments should be independent assessments in order to be objective.

Software Process Assessment Cycle

According to Paulk and colleagues (1995), the CMM-based assessment approach uses a six-step cycle. They are –

Select a team - The members of the team should be professionals knowledgeable in software engineering and management.

The representatives of the site to be appraised complete the standard process maturity questionnaire.

The assessment team performs an analysis of the questionnaire responses and identifies the areas that warrant further exploration according to the CMM key process areas.

The assessment team conducts a site visit to gain an understanding of the software process followed by the site.

The assessment team produces a list of findings that identifies the strengths and weakness of the organization's software process.

The assessment team prepares a Key Process Area (KPA) profile analysis and presents the results to the appropriate audience.

For example, the assessment team must be led by an authorized SEI Lead Assessor. The team must consist of between four to ten team members. At least, one team member must be from the organization being assessed, and all team members must complete the SEI's Introduction to the CMM course (or its equivalent) and the SEI's CBA IPI team training course. Team members must also meet some selection guidelines.

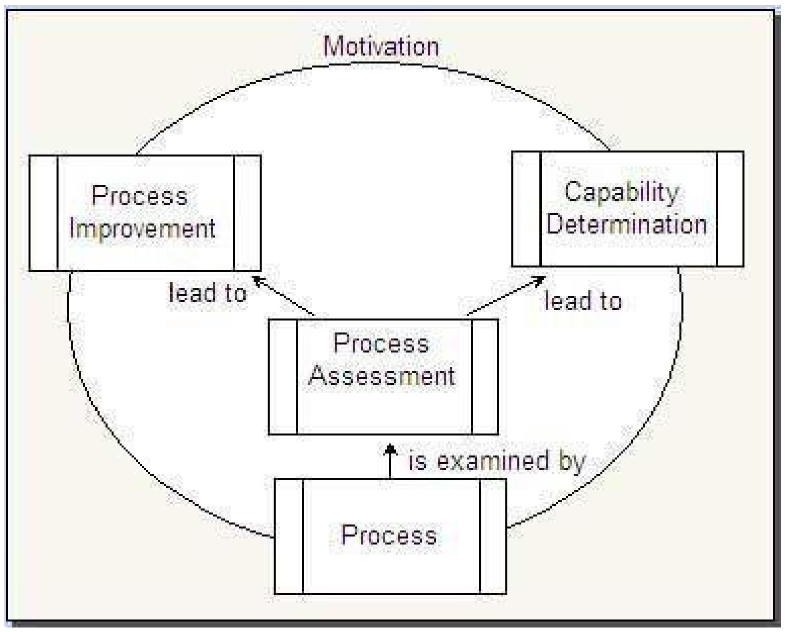
With regard to data collection, the CBA IPI relies on four methods −

The standard maturity questionnaire

Individual and group interviews

Document reviews

Feedback from the review of the draft findings with the assessment participants



SCAMPI

The Standard CMMI Assessment Method for Process Improvement (SCAMPI) was developed to satisfy the CMMI model requirements (Software Engineering Institute, 2000). It is also based on the CBA IPI. Both the CBA IPI and the SCAMPI consist of three phases −

Plan and preparation

Conduct the assessment onsite

Report results

The activities for the plan and preparation phase include −

Identify the assessment scope

Develop the assessment plan

Prepare and train the assessment team

Make a brief assessment of participants

Administer the CMMI Appraisal Questionnaire

Examine the questionnaire responses

Conduct an initial document review

The activities for the onsite assessment phase include −

Conduct an opening meeting

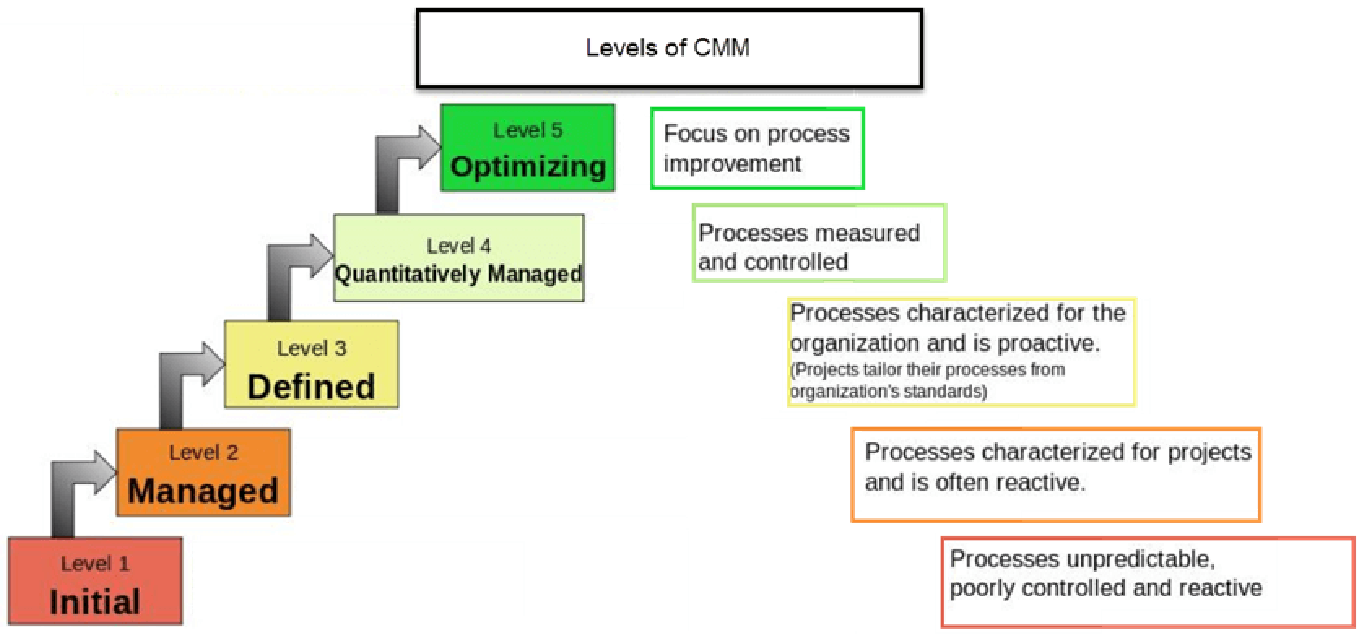
Conduct interviews

Consolidate information

Prepare the presentation of draft findings

Present the draft findings

Consolidate, rate, and prepare the final findings


Levels of CMM
Level 5
Optimizing
Focus on process improvement
Level 4
Quantitatively Managed
Processes measured and controlled
Level 3
Defined
Processes characterized for the organization and is proactive. (Projects tailor their processes from organization's standards)
Level 2
Managed
Processes characterized for projects and is often reactive.
Level 1
Initial
Processes unpredictable, poorly controlled and reactive

The Initial Process:

Initial - Work is performed informally.

- A software development organization at this level is characterized by AD HOC activities (organization is not planned in advance.)
- The software process is characterized as inconsistent, and occasionally even chaotic. Defined processes and standard practices that exist are abandoned during a crisis. Success of the organization majorly depends on an individual effort, talent, and heroics. The heroes eventually move on to other organizations taking their wealth of knowledge or lessons learnt with them.
- At the initial level, processes are disorganized, ad hoc and even chaotic. Success likely depends on individual efforts and is not considered to be repeatable. This is because processes are not sufficiently defined and documented to enable them to be replicated.
- At level 1, the process is usually chaotic and ad hoc.
- A capability is characterized on the basis of the individuals and not of the organization.
- Progress not measured
- Products developed are often schedule and over budget
- Wide variations in the schedule, cost, functionality, and quality targets

The Repeatable Process:

Level Two : Repeatable – Work is planned and tracked.

- This level of software development organization has a basic and consistent project management processes to TRACK COST, SCHEDULE, AND FUNCTIONALITY. The process is in place to repeat the earlier successes on projects with similar applications.

- This level of Software Development Organization has a basic and consistent project management processes to track cost, schedule, and functionality. The process is in place to repeat the earlier successes on projects with similar applications. Program management is a key characteristic of a level two organization.
- Requirement Management
- Estimate project parameters like cost, schedule, and functionality
- Measure actual progress
- Develop plans and process
- Software project standards are defined
- Identify and control products, problem reports changes, etc.
- Processes may differ between projects

Benefits:

- Processes become easier to comprehend
- Managers and team members spend less time in explaining how things are done and more time in executing it
- Projects are better estimated, better planned and more flexible
- Quality is integrated into projects
- Costing might be high initially but goes down overtime
- Ask more paperwork and documentation

The Defined Process:

Level Three : Defined – Work is well defined.

- At this level the software process for both management and engineering activities are DEFINED AND DOCUMENTED.
- The software process for both management and engineering activities are documented, standardized, and integrated into a standard software process for the entire organization and all projects across the organization use an approved, tailored version of the organization's standard software process for developing, testing and maintaining the application.
- Clarify customer requirements
- Solve design requirements, develop an implementation process
- Makes sure that product meets the requirements and intended use

- Analyze decisions systematically
- Rectify and control potential problems

Benefits:

- Process Improvement becomes the standard
- Solution progresses from being "coded" to being "engineered"
- Quality gates appear throughout the project effort with the entire team involved in the process
- Risks are mitigated and don't take the team by surprise

The Managed Process:

Level Four : Managed – Work is quantitatively controlled.

- **Software Quality management –** Management can effectively control the software development effort using precise measurements. At this level, organization set a quantitative quality goal for both software process and software maintenance.
- **Quantitative Process Management –** At this maturity level, The performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable.

- Management can effectively control the software development effort using precise measurements. At this level, organization set a quantitative quality goal for both software process and software maintenance.
- At this maturity level, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable.
- Manages the project's processes and sub-processes statistically
- Understand process performance, quantitatively manage the organization's project

Benefits:

- Optimizes Process Performance across the organization
- Fosters Quantitative Project Management in an organization.

The Optimizing Process:

Level Five : Optimizing

- Work is Based Upon Continuous Improvement.
  The key characteristic of this level is focusing on CONTINUOUSLY IMPROVING PROCESS performance.
- The Key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements. At this level, changes to the process are to improve the

process performance and at the same time maintaining statistical probability to achieve the established quantitative process-improvement objectives.

Key features are:

- Process change management
- Technology change management
- Defect prevention
- Detect and remove the cause of defects early
- Identify and deploy new tools and process improvements to meet needs and business objectives

Benefits:

- Fosters Organizational Innovation and Deployment
- Gives impetus to Causal Analysis and Resolution

Part -2 Process Reference Models

Capability Maturity Model(CMM)

What is CMM?

Capability Maturity Model is used as a benchmark to measure the maturity of an organization's software process.

CMM was developed at the Software engineering institute in the late 80's. It was developed as a result of a study financed by the U.S Air Force as a way to evaluate the work of subcontractors.

CMM was first introduced in late 80's in U.S Air Force to evaluate the work of subcontractors. Later on, with improved version, it was implemented to track the quality of the software development system.

The entire CMM level is divided into five levels.

Level 1 (Initial): Where requirements for the system are usually uncertain, misunderstood and uncontrolled. The process is usually chaotic and ad-hoc.

Level 2 (Managed): Estimate project cost, schedule, and functionality. Software standards are defined

Level 3 (Defined): Makes sure that product meets the requirements and intended use

Level 4 (Quantitatively Managed): Manages the project's processes and sub- processes statistically

Level 5 (Maturity): Identify and deploy new tools and process improvements to meet needs and business objectives

Why Use CMM?

Today CMM act as a "seal of approval" in the software industry. It helps in various ways to improve the software quality.

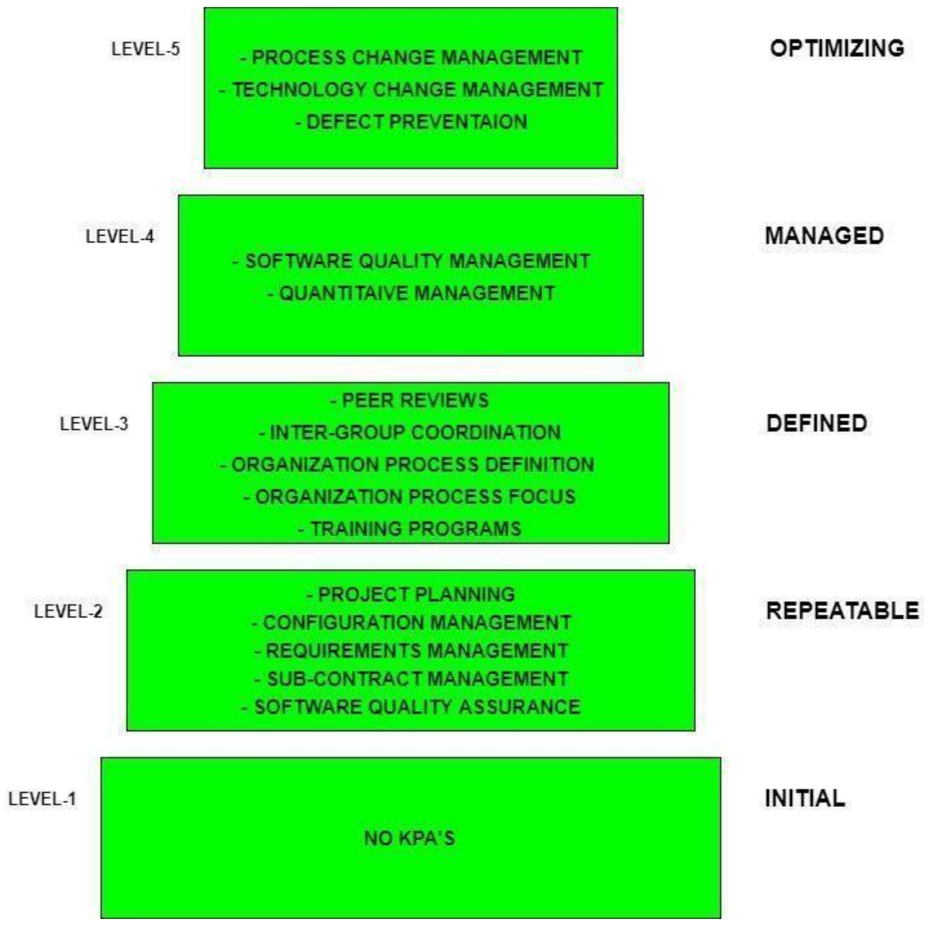
It guides towards repeatable standard process and hence reduce the learning time on how to get things done

Practicing CMM means practicing standard protocol for development, which means it not only helps the team to save time but also gives a clear view of what to do and what to expect

The quality activities gel well with the project rather than thought of as a separate event

It acts as a commuter between the project and the team

CMM efforts are always towards the improvement of the process


LEVEL-5
- PROCESS CHANGE MANAGEMENT
- TECHNOLOGY CHANGE MANAGEMENT
- DEFECT PREVENTAION
OPTIMIZING
LEVEL-4
- SOFTWARE QUALITY MANAGEMENT
- QUANTITAIVE MANAGEMENT
MANAGED
LEVEL-3
- PEER REVIEWS
- INTER-GROUP COORDINATION
- ORGANIZATION PROCESS DEFINITION
- ORGANIZATION PROCESS FOCUS
- TRAINING PROGRAMS
DEFINED
LEVEL-2
- PROJECT PLANNING
- CONFIGURATION MANAGEMENT
- REQUIREMENTS MANAGEMENT
- SUB-CONTRACT MANAGEMENT
- SOFTWARE QUALITY ASSURANCE
REPEATABLE
LEVEL-1
NO KPA'S
INITIAL

Limitations of CMM Models

CMM determines what a process should address instead of how it should be I implemented

It does not explain every possibility of software process improvement.

It concentrates on software issues but does not consider strategic business planning, adopting technologies, establishing product line and managing human resources.

It does not tell on what kind of business an organization should be in.

CMM will not be useful in the project having a crisis right now.

Capability Maturity Model Integration (CMMI)

Capability maturity model integration (CMMI) is an approach or methodology for improving and refining the software development process within an organization. It is based on a process model or a structured collection of practices.

CMMI is used to guide the improvement process across a project, division or even an entire organizational structure. It also allows companies to integrate organizational functions that are traditionally separate, set goals for process improvements and priorities, provide guidance for quality processes, and act as a point of reference for appraising processes.

Difference between CMM and CMMI

1. CMM came first but was later improved and was succeeded by CMMI.
2. Different sets of CMMS have problems with overlaps, contradictions, and lack of standardization. CMMI later addressed these problems.
3. Initially, CMM describes specifically about software engineering whereas CMMI describes integrated processes and disciplines as it applies both to software and systems engineering.
4. CMMI is much more useful and universal than the older CMM.

People Capability Maturity Model (PCMM)

The **People Capability Maturity Model** consists of five maturity levels. Each maturity level is an evolutionary plateau at which one or more domains of the organization's processes are transformed to achieve a new level of organizational capability. The five levels of **People CMM** are defined as follows:

1. At **PCMM Level 1**, an organization has no consistent way of performing workforce practices. Most workforce practices are applied without analysis of impact.
2. At **PCMM Level 2**, organizations establish a foundation on which they deploy common workforce practices across the organization. The goal of Level 2 is to have managers take responsibility for managing and developing their people. For example, the first benefit an organization experiences as it achieves Level 2 is a reduction in voluntary turnover. The turnover costs that are avoided by improved workforce retention more than pay for the improvement costs associated with achieving Level 2.
3. At **PCMM Level 3**, the organization identifies and develops workforce competencies and aligns workforce and work group competencies with business strategies and objectives. For example, the workforce practices that were implemented at Level 2 are now standardized and adapted to encourage and reward growth in the organization's workforce competencies.
4. At **PCMM Level 4**, the organization empowers and integrates workforce competencies and manages performance quantitatively. For example, the organization is able to predict its capability for performing work because it can quantify the capability of its workforce and of the competency-based processes they use in performing their assignments.
5. At **PCMM Level 5**, the organization continuously improves and aligns personal, work-group, and organizational capability. For example, at Maturity Level 5, organizations treat continuous improvement as an orderly business process to be performed in an orderly way on a regular basis.

The **People Capability Maturity Model** was designed initially for knowledge- intense organizations and workforce management processes. However, it can be applied in almost any organizational setting, either as a guide in implementing workforce improvement activities or as a vehicle for assessing workforce practices.

Personal Software Process (PSP)

The Personal Software Process (PSP) shows engineers how to

- manage the quality of their projects
- make commitments they can meet
- improve estimating and planning
- reduce defects in their products

PSP emphasizes the need to record and analyze the types of errors you make, so you can develop strategies eliminate them.

PSP model Framework Activities

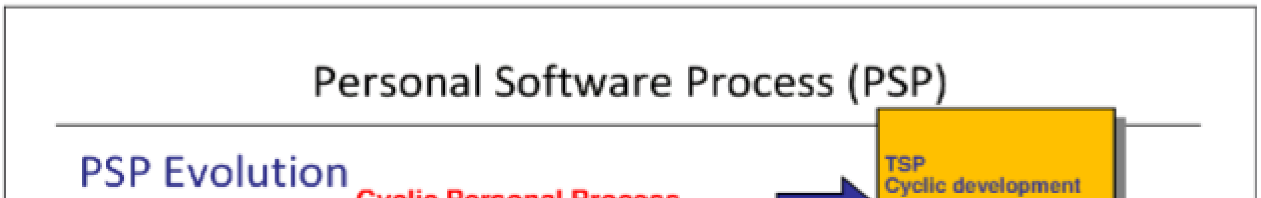
- Planning – isolates requirements and based on these develops both size & resource estimates. A defect estimate is made.
- High level Design – external specification of all components. All issues are recorded and tracked.
- High level Design Review- formal verification to uncover errors
- Development- metrics are maintained for all important tasks & work results.
- Postmortem- using measures & metrics collected effectiveness of process is determined an improved.

Because personnel costs constitute 70 percent of the cost of software development, the skills and work habits of engineers largely determine the results of the software development process.

Based on practices found in the CMMI, the PSP can be used by engineers as a guide to a disciplined and structured approach to developing software. The PSP is a prerequisite for an organization planning to introduce the TSP.

The PSP can be applied to many parts of the software development process, including

- small-program development
- requirement definition
- document writing
- systems tests
- systems maintenance
- enhancement of large software systems
- create secure software products
- improve process management in an organization



Team Software Process (TSP)

- The Team Software Process (TSP), along with the Personal Software Process, helps the high-performance engineer to ensure quality software products

TSP Framework Activities

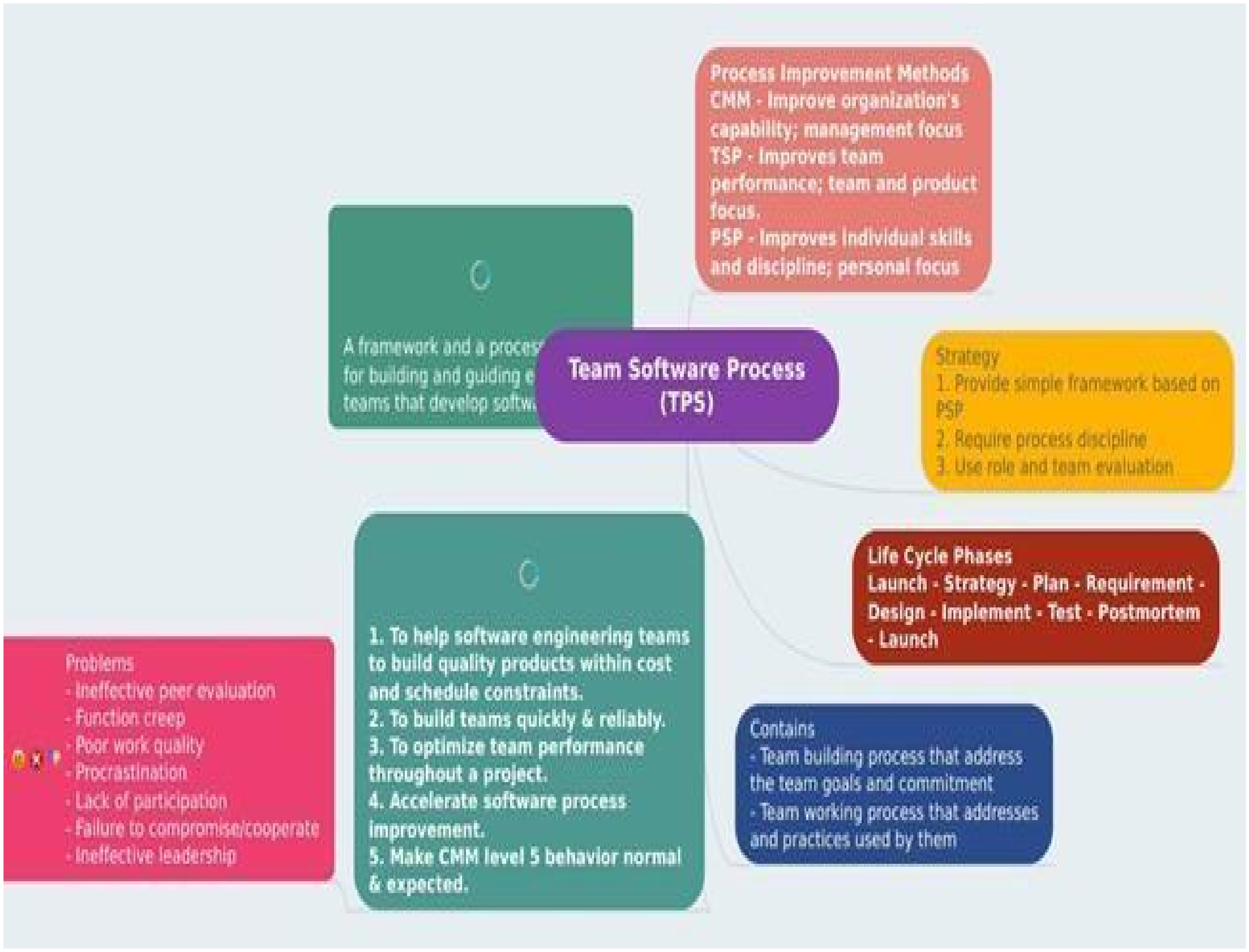
- Launch high level design
- Implementation
- Integration
- Test
- postmortem

Engineering groups use the TSP to apply integrated team concepts to the development of software-intensive systems. A launch process walks teams and their managers through

- establishing goals
- defining team roles
- assessing risks
- producing a team plan

Benefits of TSP

- The TSP provides a defined process framework for managing, tracking and reporting the team's progress.
- Using TSP, an organization can build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams of 3 to 20 engineers.
- TSP will help your organization establish a mature and disciplined engineering practice that produces secure, reliable software.


Team Software Process (TPS)
A framework and a process for building and guiding e teams that develop softw
Process Improvement Methods
CMM - Improve organization's capability; management focus
TSP - Improves team performance; team and product focus.
PSP - Improves individual skills and discipline; personal focus
Strategy
1. Provide simple framework based on PSP
2. Require process discipline
3. Use role and team evaluation
Life Cycle Phases
Launch - Strategy - Plan - Requirement - Design - Implement - Test - Postmortem - Launch
Contains
- Team building process that address the team goals and commitment
- Team working process that addresses and practices used by them
1. To help software engineering teams to build quality products within cost and schedule constraints.
2. To build teams quickly & reliably.
3. To optimize team performance throughout a project.
4. Accelerate software process improvement.
5. Make CMM level 5 behavior normal & expected.
Problems
- Ineffective peer evaluation
- Function creep
- Poor work quality
- Procrastination
- Lack of participation
- Failure to compromise/cooperate
- Ineffective leadership