Unit 1:

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).

IMPORTANT QUOTES:

If you don't know where you are going, any road will do." Chinese Proverb

"If you don't know where you are, a map won't help." Watts Humphrey

"If you don't know where you are going, a map won't get you there any faster." Anonymous

"You can't expect to be a functional employee in a dysfunctional environment" Watts Humphrey

WHY SHOULD WE MANAGE THE SOFTWARE PROCESS?

Individuals, Teams, and Armies:

History of software is one of increasing scale

Initially a few people could craft small programs

Today large projects require the coordinated work of many teams

The increase in scale requires a more structured approach to software process management

People and the Software Process

- Talented people are the most important element in a software organization
- Successful organizations provide a structured and disciplined environment to do cooperative work
- Alternative
  - Endless hours of repetitively solving technically trivial problems
  - Time is consumed by mountains of uncontrolled detail
- If the details are not managed, the best people cannot be productive
- First class people need the support of an orderly process to do first-class work

MYTH OF THE SUPER PROGRAMMERS:

- Common view: First-class people intuitively know how to do first-class work
  - Implication: No orderly process framework is needed

  - Conclusion: Organizations with the best people should not suffer from software quality and productivity problems
- However, studies show that companies with top graduates from leading universities are still plagued with the same problems
  - New Conclusion: The best people need to be supported with an effectively managed software process

MYTH OF TOOLS AND TECHNOLOGY:

- Common View: Some technically advanced tool or method will provide a magic answer to the software crisis
- Reality: Technology is vital, but unthinking reliance on an undefined "silver bullet" will divert attention from the need for better process management

MAJOR CONCERNS OF SOFTWARE PROFESSIONALS:

- Open-ended requirements
- Uncontrolled change
- Arbitrary schedules
- Insufficient test time
- Inadequate training
- Unmanaged system standards

LIMITING FACTORS IN USING SOFTWARE TECHNOLOGY:

- Poorly-defined process
- Inconsistent implementation
- Poor process management

FOCUSING ON SOFTWARE PROCESS MANAGEMENT:

- Software process: the set of actions required to efficiently transform a user's need into an effective software solution
- Many software organizations have trouble defining and controlling this process
  - Even though this is where they have the greatest potential for improvement
- This is the focus of the book "Managing the Software Process"

A SOFTWARE MATURITY FRAMEWORK:

Software maturity Framework: Fundamentally, software development must be predictable. The software process is the set of tools, methods, and practices we use to produce a software product. The objectives of software process management are to produce products according to plan while simultaneously improving the organization's capability to produce better products. The basic principles are those of statistical process control. A process is said to be stable or under statistical control if its future performance is predictable within established statistical limits.

When a process is under statistical control, repeating the work in roughly the same way will produce roughly the same result. To obtain consistently better results, it is necessary to improve the process. If the process is not under statistical control, sustained progress is not possible until it is.

Lord Kelvin - "When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced the stage of science." (But, your numbers must be reasonably meaningful.)

The mere act of measuring human processes changes them because of people's fears, and so forth. Measurements are both expensive and disruptive; overzealous measurements can disrupt the process under study.

Principles of Software Process Change:

People:

•The best people are always in short supply

•you probably have about the best team you can get right now.

•With proper leadership and support, most people can do much better than they are currently doing

Design:

•Superior products have superior design. Successful products are designed by people who understand the application (domain engineer).

•A program should be viewed as executable knowledge. Program designers should have application knowledge.

The Six Basic Principles of Software Process Change:

•Major changes to the process must start at the top

. •Ultimately, everyone must be involved.

•Effective change requires great knowledge of the current process

•Change is continuous

•Software process changes will not be retained without conscious effort and periodic reinforcement

•Software process improvement requires investment.

Continuous Change:

•Reactive changes generally make things worse

•Every defect is an improvement opportunity

•Crisis prevention is more important than crisis recovery

SOFTWARE PROCESSES CHANGES WON'T STICK BY THEMSELVES

The tendency for improvements to deteriorate is characterized by the term entrophy (Webster's: a measure of the degree of disorder in a...system; entrophy always increases and available energy diminishes in a closed system.). New methods must be carefully introduced and periodically monitored, or they to will rapidly decay. Human adoption of new process involves four stages:

• Installation - Initial training

• Practice - People learn to perform as instructed

• Proficiency - Traditional learning curve

• Naturalness - Method ingrained and performed without intellectual effort.

It Takes Time, Skill, and Money!

•To improve the software process, someone must work on it

•Unplanned process improvement is wishful thinking

•Automation of a poorly defined process will produce poorly defined results

•Improvements should be made in small steps

•Train!!!!

Some Common Misconceptions about the Software Process

•We must start with firm requirements

•If it passes test it must be OK

•Software quality can't be measured

•The problems are technical

•We need better people

•Software management is different

SOFTWARE PROCESS ASSESSMENT

Process assessments help software organizations improve themselves by identifying their crucial problems and establishing improvement priorities. The basic assessment objectives are:

•Learn how the organization works

•Identify its major problems

•Enroll its opinion leaders in the change process

The essential approach is to conduct a series of structured interviews with key people in the organization to learn their problems, concerns, and creative ideas.

ASSESSMENT OVERVIEW:

A software assessment is not an audit. Audits are conducted for senior managers who suspect problems and send in experts to uncover them. A software process assessment is a review of a software organization to advise its management and professionals on how they can improve their operation.

The phases of assessment are:

•Preparation - Senior management agrees to participate in the process and to take actions on the resulting recommendations or explain why not. Concludes with a training program for the assessment team

•Assessment - The on-site assessment period. It takes several days to two or more weeks. It concludes with a preliminary report to local management.

•Recommendations - Final recommendations are presented to local managers. A local action team is then formed to plan and implement the recommendations.

Five Assessment Principles:

•The need for a process model as a basis for assessment

•The requirement for confidentiality

•Senior management involvement

•An attitude of respect for the views of the people in the organization be assessed

•An action orientation

Start with a process model - Without a model, there is no standard; therefore, no measure of change. Observe strict confidentiality - Otherwise, people will learn they cannot speak in confidence. This means managers can't be in interviews with their subordinates. Involve senior management - The senior manager (called site manager here) sets the organizations priorities. The site manager must be personally involved in the assessment and its follow-up actions. Without this support, the assessment is a waste of time because lasting improvement must survive periodic crises. Respect the people in the assessed organization - They probably work hard and are trying to improve. Do not appear arrogant; otherwise, they will not cooperate and may try to prove the team is ineffective. The only source of real information is from the workers.

Assessment recommendations should highlight the three or four items of highest priority. Don't overwhelm the organization. The report must always be in writing. Implementation Considerations - The greatest risk is that no significant improvement actions will be taken (the "disappearing problem" syndrome). Superficial changes won't help. A small, full-time group should guide the implementation effort, with participation from other action plan working groups. Don't forget that site managers can change or be otherwise distracted, so don't rely on that person solely, no matter how committed.

THE INITIAL PROCESS(LEVEL1)

Usually ad hoc and chaotic - Organization operates without formalized procedures, cost estimates, and project plans. Tools are neither well integrated with the process nor uniformly applied. Change control is lax, and there is little senior management exposure or understanding of the problems and issues. Since many problems are deferred or even forgotten, software installation and maintenance often present serious problems. While organizations at this level may have formal procedures for planning and tracking work, there is no management mechanism to insure they are used. Procedures are often abandoned in a crisis in favor of coding and testing. Level 1 organizations don't use design and code inspections and other techniques not directly related to shipping a product. Organizations at Level 1 can improve their performance by instituting basic project controls.

The most important ones are

•Project management

•Management oversight

•Quality assurance

•Change control

THE REPEATABLE PROCESS (LEVEL 2)

This level provides control over the way the organization establishes plans and commitments. This control provides such an improvement over Level 1 that the people in the organization tend to believe they have mastered the software problem. This strength, however, stems from their prior experience in doing similar work. Level 2 organizations face major risks when presented with new challenges.

Some major risks:

•New tools and methods will affect processes, thus destroying the historical base on which the organization lies. Even with a defined process framework, a new technology can do more harm than good.

•When the organization must develop a new kind of product, it is entering new territory.

•Major organizational change can be highly disruptive. At Level 2, a new manager has no orderly basis for understanding an organization's operation, and new members must learn the ropes by word of mouth. Key actions required to advance from Repeatable to the next stage, the Defined Process, are:

•Establish a process group: A process group is a technical resource that focuses heavily on improving software processes. In most software organizations, all the people are generally devoted to product work. Until some people are assigned full-time to work on the process, little orderly progress can be made in improving it.

•Establish a software development process architecture (or development cycle) that describes the technical and management activities required for proper execution of the development process. The architecture is a structural decomposition of the development cycle into tasks, each of which has a defined set of prerequisites, functional decompositions, verification procedures, and task completion specifications.

•Introduce a family of software engineering methods and technologies. These include design and code inspections, formal design methods, library control systems, and comprehensive testing methods. Prototying and modern languages should be considered.

THE DEFINED PROCESS (LEVEL 3)

The organization has the foundation for major and continuing change. When faced with a crisis, the software teams will continue to use the same process that has been defined.

However, the process is still only qualitative; there is little data to indicate how much is accomplished or how effective the process is. There is considerable debate about the value of software process measurements and the best one to use.

The key steps required to advance from the Defined Process to the next level are:

•Establish a minimum set of basic process measurements to identify the quality and cost parameters of each process step. The objective is to quantify the relative costs and benefits of each major process activity, such as the cost and yield of error detection and correction methods.

•Establish a process database and the resources to manage and maintain it. Cost and yield data should be maintained centrally to guard against loss, to make it available for all projects, and to facilitate process quality and productivity analysis. Provide sufficient process resources to gather and maintain the process data and to advise project members on its use. Assign skilled professionals to monitor the quality of the data before entry into the database and to provide guidance on the analysis methods and interpretation.

•Assess the relative quality of each product and inform management where quality targets are not being met. Should be done by an independent quality assurance group.

THE MANAGED PROCESS (LEVEL 4)

Largest problem at Level 4 is the cost of gathering data. There are many sources of potentially valuable measure of the software process, but such data are expensive to collect and maintain.

Productivity data are meaningless unless explicitly defined. For example, the simple measure of lines of source code per expended development month can vary by 100 times or more,

depending on the interpretation of the parameters When different groups gather data but do not use identical definitions, the results are not comparable, even if it makes sense to compare them. It is rare when two processes are comparable by simple measures. The variations in task complexity

caused by different product types can exceed five to one. Similarly, the cost per line of code for small modifications is often two to three times that for new programs.

Process data must not be used to compare projects or individuals. Its purpose is too illuminate the product being developed and to provide an informed basis for improving the process.

When such data are used by management to evaluate individuals or terms, the reliability of the data itself will deteriorate. The two fundamental requirements for advancing from the Managed Process to the next level are:

•Support automatic gathering of process data. All data is subject to error and omission, some data cannot be gathered by hand, and the accuracy of manually gathered data is often poor.

•Use process data to analyze and to modify the process to prevent problems and improve efficiency.

THE OPTIMIZING PROCESS (LEVEL 5)

To this point software development managers have largely focused on their products and will typically gather and analyze only data that directly relates to product improvement. In the Optimizing Process, the data are available to tune the process itself. For example, many types of errors can be identified far more economically by design or code inspections than by testing. However, some kinds of errors are either uneconomical to detect or almost impossible to find except by machine. Examples are errors involving interfaces, performance, human factors, and error recovery.

So, there are two aspects of testing: removal of defects and assessment of program quality. To reduce the cost of removing defects, inspections should be emphasized. The role of functional and system testing should then be changed to one of gathering quality data on the program. This involves studying each bug to see if it is an isolated problem or if it indicates design problems that require more comprehensive analysis. With Level 5, the organization should identify the weakest elements of the process and fix them. Data are available to justify the application of technology to various critical tasks, and numerical evidence is available on the effectiveness with which the process has been applied to any given product.

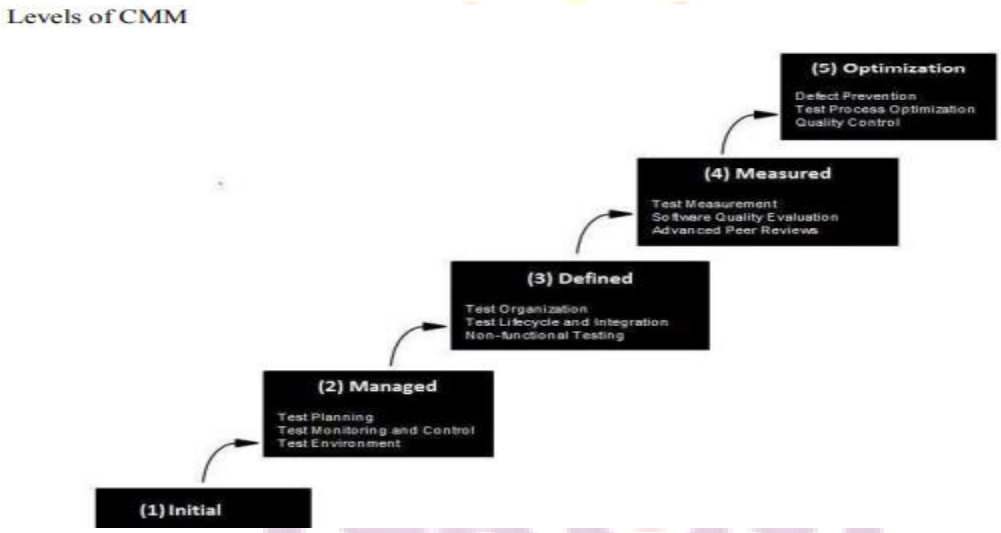
Process reference models; The process framework or reference model acts as an interface between the way the content is organized and the way work is performed. A uniform process model organized under a process reference model makes business modeling and systems designing much

easier

CAPABILITY MATURITY MODEL (CMM):

Broadly refers to a process improvement approach that is based on a process model. CMM also refers specifically to the first such model, developed by the Software Engineering Institute (SEI) in the mid-1980s, as well as the family of process models that followed.

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and timeconsuming process.



Level One :Initial - 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.

Level Two: Repeatable - 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.

Level Three: Defined - 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.

Level Four: Managed - Management can effectively control the softwaredevelopment 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.

Level Five: Optimizing - 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.

WHAT IS CMMI ?

CMM Integration project was formed to sort out the problem of using multiple CMMs.

CMMI Product Team's mission was to combine three Source Models into a single improvement framework to be used by the organizations pursuing enterprise-wide process improvement. These three Source Models are :

- Capability Maturity Model for Software (SW-CMM) - v2.0 Draft C
- Electronic Industries Alliance Interim Standard (EIA/IS) - 731 Systems Engineering
- Integrated Product Development Capability Maturity Model (IPD-CMM) v0.98

CMM Integration:

- builds an initial set of integrated models.
- - improves best practices from source models based on lessons learned.
- - establishes a framework to enable integration of future models.

Following are obvious objectives of CMMI:

Produce quality products or services: The process-improvement concept in CMMI models evolved out of the Deming, Juran, and Crosby quality paradigm: Quality products are a result of quality processes. CMMI has a strong focus on qualityrelated activities including requirements

management, quality assurance, verification, and validation.

Create value for the stockholders: Mature organizations are more likely to make better cost and revenue estimates than those with less maturity, and then perform in line with those estimates. CMMI supports quality products, predictable schedules, and effective measurement to support management in making accurate and defensible forecasts. This process maturity can guard against project performance problems that could weaken the value of the organization in the eyes of investors.

Enhance customer satisfaction: Meeting cost and schedule targets with high-quality products that are validated against customer needs is a good formula for customer satisfaction. CMMI addresses all of these ingredients through its emphasis on planning, monitoring, and measuring, and the improved predictability that comes with more capable processes.

The CMM Integration is a model that has integrated several disciplines/bodies of knowledge. Currently there are four bodies of knowledge available to you when selecting a CMMI model.

SYSTEMS ENGINEERING

Systems engineering covers the development of complete systems, which may or may not include software. Systems engineers focus on transforming customer needs, expectations, and constraints into product solutions and supporting these product solutions throughout the entire lifecycle of the product.

SOFTWARE ENGINEERING

Software engineering covers the development of software systems. Software engineers focus on the application of systematic, disciplined, and quantifiable approaches to the development, operation, and maintenance of software.

INTEGRATED PRODUCT AND PROCESS DEVELOPMENT

Integrated Product and Process Development (IPPD) is a systematic approach that achieves a timely collaboration of relevant stakeholders throughout the life of the product to better satisfy customer needs, expectations, and requirements. The processes to support an IPPD approach are integrated with the other processes in the organization. If a project or organization chooses IPPD, it performs

the IPPD best practices concurrently with other best practices used to produce products (e.g., those related to systems engineering). That is, if an organization or project wishes to use IPPD, it must select one or more disciplines in addition to IPPD.

SUPPLIER SOURCING

As work efforts become more complex, project managers may use suppliers to perform functions or add modifications to products that are specifically needed by the project. When those activities are critical, the project benefits from enhanced source analysis and from monitoring supplier activities before product delivery. Under these circumstances, the supplier sourcing discipline covers the acquisition of products from suppliers. Similar to IPPD best practices, supplier sourcing best practices must be selected in conjunction with best practices used to produce products.

CMMI Discipline Selection Selecting a discipline may be a difficult step and depends on what an organization wants to improve.

- If you are improving your systems engineering processes, like Configuration Management, Measurement and Analysis, Organizational Process Focus, Project Monitoring and Control, Process and Product Quality Assurance, Risk Management, Supplier Agreement Management etc., then you should select Systems engineering (SE) discipline. The discipline amplifications for systems engineering receive special emphasis.
- If you are improving your integrated product and process development processes like Integrated Teaming, Organizational Environment for Integration, then you should select IPPD. The discipline amplifications for IPPD receive special emphasis.
- If you are improving your source selection processes like Integrated Supplier Management then you should select Supplier sourcing (SS). The discipline amplifications for supplier sourcing receive special emphasis.
- If you are improving multiple disciplines, then you need to work on all the areas related to those disciplines and pay attention to all of the discipline amplifications for those disciplines

The CMMI is structured as follows −

 Maturity Levels (staged representation) or Capability Levels (continuous representation)

- Process Areas
- Goals: Generic and Specific
- Common Features
- Practices: Generic and Specific

This chapter will discuss about two CMMI representations and rest of the subjects will be covered in subsequent chapters.

A representation allows an organization to pursue different improvement objectives. An organization can go for one of the following two improvement paths.

Staged Representation The staged representation is the approach used in the Software CMM. It is an approach that uses predefined sets of process areas to define an improvement path for an organization. This improvement path is described by a model component called a Maturity Level. A maturity level is a well-defined evolutionary plateau towards achieving improved organizational processes.

CMMI Staged Representation Provides a proven sequence of improvements, each serving as a foundation for the next. Permits comparisons across and among organizations by the use of maturity levels. Provides an easy migration from the SW-CMM to CMMI. Provides a single rating that summarizes appraisal results and allows comparisons among organizations.

Thus Staged Representation provides a pre-defined roadmap for organizational improvement based on proven grouping and ordering of processes and associated organizational relationships. You cannot divert from the sequence of steps.

The People Capability Maturity Model (People CMM, P-CMM) is part of the CMMI product family of process maturity models. It is a framework to guide organisations in improving their processes for managing and developing human workforces. It helps organisations to characterize the maturity of their workforce practices, establish a program of continuous workforce development, set priorities for improvement actions, integrate workforce development with Process Improvement, and establish a culture of excellence. PCMM is

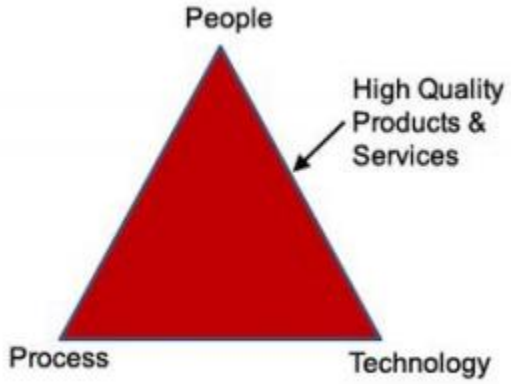
based on proven practices in fields of human resources, knowledge management, and organisational development. P-CMM is part of the CMMI product family of process maturity models. It describes a progression for continuous improvement and process improvement of the HR processes for managing and developing human workforces. The P-CMM framework enables organisations to incrementally focus on key process areas and to lay foundations for improvement in workforce practices. Unlike other HR models, P-CMM requires that key process areas, improvements, interventions, policies, and procedures are institutionalised across the organisation — irrespective of function or level. Therefore, all improvements have to percolate throughout the organisation, to ensure consistency of focus, to place emphasis on a participatory culture, embodied in a team-based environment, and encouraging individual innovation and creativity. Process Maturity Rating The process maturity rating is from ad hoc and inconsistently performed practices, to a mature and disciplined development of the knowledge, skills, and motivation of the workforce.

Traditionally, process maturity models like ISO/IEC 15504 or CMMI focus on organisational improvement with respect to operational (Product) Development Processes. PCMM in contrast focus instead on the three prominent factors for operational capability to deliver successful products and services:

1. People

2. Process

3. Products, Technology



Levels	People CMM Threads			
	Developing competency	Building workgroups& culture	Motivating & managing performance	Shaping the workforce
5 Optimizing	Continuous Capability Improvement		Organisational Performance Alignment	Continuous Workforce Innovation
4 Predictable	Competency Based Assets Mentoring	Competency Integration Empowered workgroups	Quantitative Performance Management	Organisational Capability Management
3 Defined	Competency development Competency Analysis	Workgroup Development Participatory Culture	Competency Based Practices Career Development	Workforce Planning
2 Managed	Training and Development	Communication & Coordination	Compensation Performance Management	Staffing

PSP

The Personal Software Process (PSP) is a structured software development process that is designed to help software engineers better understand and improve their performance by bringing discipline to the way they develop software and tracking their predicted and actual development of the code. It clearly shows developers how to manage the quality of their products, how to make a sound plan, and how to make commitments. It also offers them the data to justify their plans. They can evaluate their work and suggest improvement direction by analyzing and reviewing development time, defects, and size data. The PSP was created by Watts Humphrey to apply the underlying principles of the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) to the software development practices of a single developer. It claims to give software engineers the process skills necessary to work on a team software process (TSP) team.

The PSP aims to provide software engineers with disciplined methods for improving personal

software development processes. The PSP helps software engineers to:

 Improve their estimating and planning skills.

 Make commitments they can keep.

 Manage the quality of their projects.

 Reduce the number of defects in their work.

PSP training follows an evolutionary improvement approach: an engineer learning to integrate the PSP into his or her process begins at the first level – PSP0 – and progresses in process maturity to the final level – PSP2.1. Each Level has detailed scripts, checklists and templates to guide the engineer through required steps and helps the engineer improve their own personal software process. Humphrey encourages proficient engineers to customize these scripts and templates as they gain an understanding of their own strengths and weaknesses.

 Process The input to PSP is the requirements; requirements document is completed and delivered to the engineer.

TSP

The team software process (TSP) provides a defined operational process framework that is designed to help teams of managers and engineers organize projects and produce software the principles products that range in size from small projects of several thousand lines of code (KLOC) to very large projects greater than half a million lines of code. The TSP is intended to improve the levels of quality and productivity of a team's software development project, in order to help them better meet the cost and schedule commitments of developing a software system The initial version of the TSP was developed and piloted by Watts Humphrey in the late 1990s and the Technical Report for TSP sponsored by the U.S. Department of Defense was published in November 2000. The book by Watts Humphrey, Introduction to the Team Software Process, presents a view of the TSP intended for use in academic settings, that focuses on the process of building a software production team, establishing team goals, distributing team roles, and other teamwork-related activities. The primary goal of TSP is to create a team environment for establishing and maintaining a self-directed team, and supporting disciplined individual work as a base of PSP framework. Self-directed team means that the team

manages itself, plans and tracks their work, manages the quality of their work, and works proactively to meet team goals. TSP has two principal components: team-building and team-working. Team-building is a process that defines roles for each team member and sets up teamwork through TSP launch and periodical relaunch. Team-working is a process that deals with engineering processes and practices utilized by the team. TSP, in short, provides engineers and managers with a way that establishes and manages their team to produce the high-quality software on schedule and budget.

HOW TSP WORKS:

Before engineers can participate in the TSP, it is required that they have already learned about the PSP, so that the TSP can work effectively. Training is also required for other team members, the team lead and management. The TSP software development cycle begins with a planning process called the launch, led by a coach who has been specially trained, and is either certified or provisional. The launch is designed to begin the team building process, and during this time teams and managers establish goals, define team roles, assess risks, estimate effort, allocate tasks, and produce a team plan. During an execution phase, developers track planned and actual effort, schedule, and defects meeting regularly (usually weekly) to report status and revise plans. A development cycle ends with a Post Mortem to assess performance,revise planning parameters, and capture lessons learned for process improvement. The coach role focuses on supporting the team and the individuals on the team as the process expert while being independent of direct project management responsibility. The team leader role is different from the coach role in that, team leaders are responsible to management for products and project outcomes while the coach is responsible for developing individual and team performance