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).

1.1 Software Process

A software process is a set of related activities that leads to the production of a software product. These activities may involve the development of software from scratch in a standard programming language like Java or C. However, business applications are not necessarily developed in this way. New business software is now often developed by extending and modifying existing systems or by configuring and integrating off-the-shelf software or system components. There are many different software processes but all must include four activities that are fundamental to software engineering.

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

1.1.1 Software Process Improvement

To improve their software capabilities, organizations must take six steps;

- •Understand the current status of the development processes
- •Develop a vision of the required process
- •Establish a list of required process improvement actions in order of priority
- •Produce a plan to accomplish the required action
- •Commit the resources to execute the plan
- •Start over at the first step.

1.2 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 meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced the stage of science.”

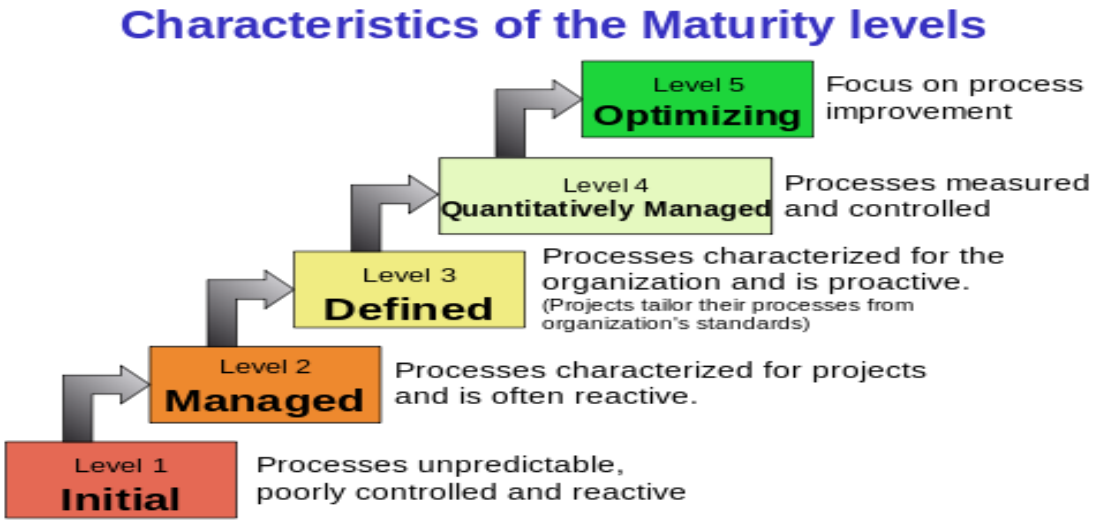

Figure 1: Characteristics of Maturity Level

1.2.1 Process Maturity Levels

1. **Initial.** Until the process is under statistical control, orderly progress in process improvement is not possible. Must at least achieve rudimentary predictability of schedules and costs.

2. **Repeatable**. The organization has achieved a stable process with a repeatable level of statistical control by initiating rigorous project management of commitments, costs, schedules, and changes.

3. **Defined.** The organization has defined the process as a basis for consistent implementation and better understanding. At this point, advanced technology can be introduced.

4. **Managed**. The organization has initiated comprehensive process measurements and analysis. This is when the most significant quality improvements begin.

5. **Optimizing**. The organization now has a foundation for continuing improvement and optimization of the process.

These levels are selected because they

- •Represent the historical phases of evolutionary improvement of real software organizations
- •Represent a measure of improvement that is reasonable to achieve from a prior level
- •Suggest improvement goals and progress measures
- •Make obvious a set of intermediate improvement priorities once an organization’s status in this framework is known.

The Initial Process (Level 1)

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 favour 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 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. Prototyping 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.

1.3 The 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.

1.3.1 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 *entropy* (Webster's: a measure of the degree of disorder in a...system; entropy 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.

Firm Requirements

A software perversity law seems to be the more firm the specifications, the more likely they are to be wrong. With rare exceptions, requirements change as the software job progresses. Just by writing a program, we change our perceptions of the task. Requirements cannot be firm because we cannot anticipate the ways the tasks will change when they are automated.

For large-scale programs, the task of stating a complete requirement is not just difficult; it is impossible. Generally, we must develop software incrementally.
However, we must have stability long enough to build and test a system. However, if we freeze requirements too early, later retrofits are expensive.

Software Management Is Different

Management must not view it as black art. Must insist on tracking plans and reviews.

Champions, Sponsors, and Agents

•Champions - Ones who initiate change. They bring management's attention to the subject, obtain the blessing of a sponsor, and establish the credibility to get the change program launched. The champion maintains focus on the goal, strives to overcome obstacles, and refuses to give up when the going gets tough.

•Sponsors - Senior manager who provides resources and official backing. Once a sponsor is found, the champion's job is done; it is time to launch the change process.

•Agents - Change agents lead the planning and implementation. Agents must be enthusiastic, technically and politically savvy, respected by others, and have management's confidence and support.

Elements of Change

•Planning
•Implementation
•Communication

1.4 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. Audit 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. 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.

1.5 The Initial Process

In the Initial Process, professionals are driven from crisis to crisis by unplanned priorities and unmanaged change. Such groups seldom meet commitments. They may deliver on schedule occasionally, but this is the exception and normally due to herculean individual

efforts rather than the strength of the organization. Senior management makes unreasonable demands while cutting resources. Schedules are the only priority, but when they do deliver, nobody likes the results. Many first-class professionals leave as soon as they can. Most of the rest have strong local ties or have just given up and collect their pay.

Customers tolerate such organizations only if there are no alternatives.

Why Organizations Are Chaotic(Disorganized)

Simplest explanation: People don't like to admit they don't know. When political pressure amounts for a commitment, it is difficult to say you don't know. So, people tend to promise products they can't deliver.

While lack of a commitment discipline is the common reason for chaotic behavior, there are other forces:

- •Under extreme pressure, software managers make a guess instead of a plan. The guess is usually low, so chaos develops.
- •When things get rough, there may be a strong temptation to believe in magic. A savior or new technology may be the answer.
- •The scale of software projects normally follows an escalating cycle.
- •Programs take more code than expected;
- •As the programs become larger, new technical and management issues arise.
- •Since these are unlike previous experience, they are a surprise.
- •Even after a higher maturity level is reached, new management, increased competition, or new technical challenges put pressure on processes; then, an organization may revert to the Initial Process.

Unplanned Commitments

When a new feature seems simple, management may just commit to do it without planning. Often, the change is far more complicated than thought. This results in confusion, changed priorities, and stress. The users and senior management become frustrated and unwilling to listen to excuses when this becomes a pattern.

Gurus

Gurus can make this worse because they run projects out of their heads. With nothing written down, everyone comes to them for guidance. At the breaking point, the limits of this intuitive approach have been reached and there is no simple recovery.

Magic

"No silver bullets" - Brooks

PROBLEMS OF SCALE

Software size is insidious because of its impact on the development process. Building small programs does not lead to discipline needed to build large software systems.
As software products become larger, the progressive level of scale are roughly as follows:

•One person knows the details of the program.
•One person understands it but can't remember it so the design must be documented
•One person understands the overall design of the program, but the details of its component modules are each understood by separate experts.
•A large software project is defined and understood at the product management level, but a separate team understands the characteristics and design of each of its component programs.
•With software systems, the high-level design may be well defined and understood by the system management team, but each of the component products is only understood by the respective product management organizations.
•When the system is very large and has evolved through many versions, there may be no one who understands it.

As software knowledge is more widely distributed:

•Common notations are needed for precise communication. These standards must be documented, interpreted, and updated.
•Conflicts in standards must be identified and resolved
•Standards changes must be controlled and distributed

With large-scale software, similar control is needed for requirements, design, code, and test. As software size increases, prototypes or multiple releases are needed because:

•The total function cannot be implemented at one time.
•Some needs cannot be understood without operational experience on a partial system.
•Some design issues cannot be resolved until a preliminary system has been built and run.
•A release discipline helps sort out user priorities.
•No one successfully builds software in one shot, anyway.

With multiple releases, new complications arise:
•The requirements must be phased to meet end user needs.
•Each software component design must synchronize with these needs.
•Product interdependencies are orchestrated to meet release functional requirements
•The build and integration plans are scheduled around these interdependencies
•Early system drivers are scheduled to meet component test needs.
•The tight schedule requires subsequent release development to start before the prior releases are finished.

Software Process Entrophy

As we build new systems, we learn what we should have built. This is a good reason for prototypes. This can help avoid chaos.

The Way Out

•Apply systematic project management - the work must be estimated, planned, and managed.

•Adhere to careful change control - changes must be controlled, including requirements, design, implementation, and test.
•Utilize independent software assurance - an independent technical means is required to assure that all essential project activities are properly performed.

The basic principles for controlling chaos in software organizations are:

•Plan the work
•Track and maintain the plan
•Divide the work into individual parts
•Precisely define the requirements for each part
•Rigorously control the relationships among the parts
•Treat software development as a learning process
•Recognize what you don't know
•When the gap between your knowledge and the task is severe, fix it before proceeding
•Manage, audit, and review the work to ensure it is done as planned
•Commit to your work and work to your commitments
•Refine the plan as your knowledge improves.

1.6 Repeatable Process

Managing Software Organizations

The basic principles of project management are:
•Each project has a plan that is based on a hierarchy of commitments;
•A management system resolves the natural conflicts between the projects and between the line and staff organizations;
•An oversight and review system audits and tracks progress against the plans.

The foundation for software project management is the commitment discipline. Commitments are supported by plans, reviews, and so forth, but commitments are met by people.

The elements of an effective commitment are:

•The person making the commitment does so willingly;
•The commitment is not made lightly;
•There is agreement between the parties on what is to be done, by whom, and when;
•The commitment is openly stated;
•The person tries to meet the commitment, even if help is needed;
•Prior to the commitment date, if it is clear that it cannot be met, advance notice is given and a new commitment is negotiated.

The software commitment process:

•All commitments for future software delivery are made personally by the organization's senior executive;
•These commitments are made only after successful completion of a formal review and concurrence process.
•An enforcement mechanism ensures these reviews and concurrences are properly conducted.

The senior manager should require evidence that the following work was done prior to approving a commitment:

•The work has been defined between the developers and the customer;

•A documented plan has been produced, including a resource estimate, a schedule, and a cost estimate.

•All directly involved parties have agreed to this plan in writing.

•Adequate planning has been done to ensure the commitment is a reasonable risk.

•An independent review ensures the planning work was done according to the organization's standards and procedures.

•The groups doing the work have or can acquire the resources needed.

Product and Period Plans

The period (operating) plan deals with technical and business issues in annual and organizational terms. Thus expenses, capital requirements, and product delivery commitments are established for each period by each organizational entity.

Product plans focus on the activities and objectives of each product. The issues are function, cost, schedule, and quality, together with related resources and checkpoints.

The distinction between period and product can be confusing. Project personnel view their work as the fundamental business of the organization and have little appreciation for period information. The project is not set up on an annual basis and often has considerable difficulty in producing annual data on items such as cost, quality, or productivity.

Organizations, however, are generally measured on a period basis, and the project must be translated into period terms for inclusion in annual budgets, plans, or stockholders reports.

The organization's direction is set in its strategy. The strategy is a period instrument that deals with the problems and issues five or more years ahead. It builds the framework for continual organizational improvement.

The Contention Process

An effective management system requires a parallel contention system to encourage the open expression of differences and their rational resolution. The principle is that the best decisions are based on full understanding of the relevant issues.

The principles of the contention system are:

•All major decisions are reviewed with the involved parties in advance, and the parties are requested to agree. Where possible, any issues are resolved before proceeding;

•When the time comes for a decision, all dissenting parties are present and asked to state their views;

•When there is no agreement, senior management determines if there is knowledgeable agreement, if any disagreeing parties are absent, or if more preparation is needed. In the latter two cases, the decision is deferred until the necessary homework is done.

This system does not generate dissention that does not already exist, but it does reveal that dissention and optimize its utility.

Management must develop a system for tracking projects and commitments.

1.7 The Defined Process

Why define a process?

There are a lot of reasons why an organization might decide that it's time to define a process. Usually workers and management start noticing that there is a lot of time being spent not on the actual work, but on fire-fighting (e.g., doing rework, fixing, re-fixing, testing, retesting, putting out fires caused by tasks that weren't done at all, repairing relationships with customers, and running around to see if anyone remembers what worked last time). We've all seen ten people doing the "same" work without a process—no one agrees on exactly what is done, or how it should be done, when, or by whom. That the work gets done at all may be surprising. When an organization goes into fire fighter, or reactive mode, the solution often seems to be to hire more people. So now we can have twelve or fourteen people putting out fires, instead of just ten.

That's sure to improve productivity and quality. Eventually, someone may decide that it's time to turn off the sirens, do some analysis, put together a team, and get the process defined, implemented, and managed.

Some benefits of process work include

1. Increase the effectiveness of preventive measures,
2. Reduce variation, increase reliability,
3. Reduce redundant or non-critical work,
4. Reduce rework and retesting,
5. Increase the cooperation between departments or teams,
6. Improve cycle-time and quality, and
7. Improve the ability to forecast and estimate.

Process definition and management efforts ultimately affect customer satisfaction, employee morale, and the organization's profitability.

Defining A Process: Overview

The following diagram illustrates the steps to follow in defining a process. Three roles are involved: the Sponsor, the Facilitator, and the Process Definition Team. The Sponsor the Sponsor enables and motivates the Process Definition Team by mapping the organization's vision, mission, and principles to the process team's scope and goals. The Sponsor demonstrates to the team that the work they are doing is a priority by helping to remove barriers and ensuring that the team has the resources they need. The best candidate for a Sponsor is a high-level manager who is responsible for at least part of the functions in the process and can be involved with the Process Definition Team.

The Facilitator

The Facilitator leads the team through the definition, implementation, and management of the process. The Facilitator needs to have skills in group facilitation, team building, metrics and measurement theory and practice, and problem-solving tools, as well as practical experience in process definition.

The Process Definition Team

The Process Definition Team is comprised of workers from the organizations that are most involved in the process being defined. Teams must include workers from the level

"where the action is," that is, the workers who actually perform the steps in the process. The team should not have more than six primary members. If there are more than six departments or functions involved in the process, workers from groups with less involvement in the process can meet with the primary team as needed.

Reading the Process Diagram

The conventions used in the diagram "How to Define a Process" were designed to make the process flow diagram easy to read for technical and non-technical staff alike. The diagram clearly shows who is doing what, and when.
The following conventions were followed for this diagram, and are recommended in defining all processes:
1. Each role in the process has a horizontal "action zone" showing steps performed by persons in that role.
2. Steps that overlap action zones are steps performed together.
3. Steps with dotted-line borders are "ongoing" steps that continue to be done throughout the process, as needed.
4. A step with heavy, thick borders is an "external process" step, indicating a separate process being called.

The Steps In Defining A Process

The following is a condensed version of the step descriptions for each step in the process flow diagram. Information on obtaining the complete documentation follows of this paper.

Step 1—Initiate Process Definition

The Sponsor and the Facilitator work together to provide the framework for the Process Definition Team. It is the responsibility of the Facilitator to explain the benefits of process definition and implementation, how the work will be accomplished, and how much effort will be involved. The Facilitator explains to the Sponsor about the importance of having team members who actually do the work, and asks the Sponsor for a list of potential team members. The Sponsor provides information about the organization and the process to be defined.

The purpose of this step is to see that the following are fully documented:

1. Organization Vision and Mission Statement
2. Process description, and description of how the process helps with the organization's mission
3. Scope of the process. When does the process start? End? What activities, functions, or outputs are not to be included in this process? Which organizations are involved in this process?

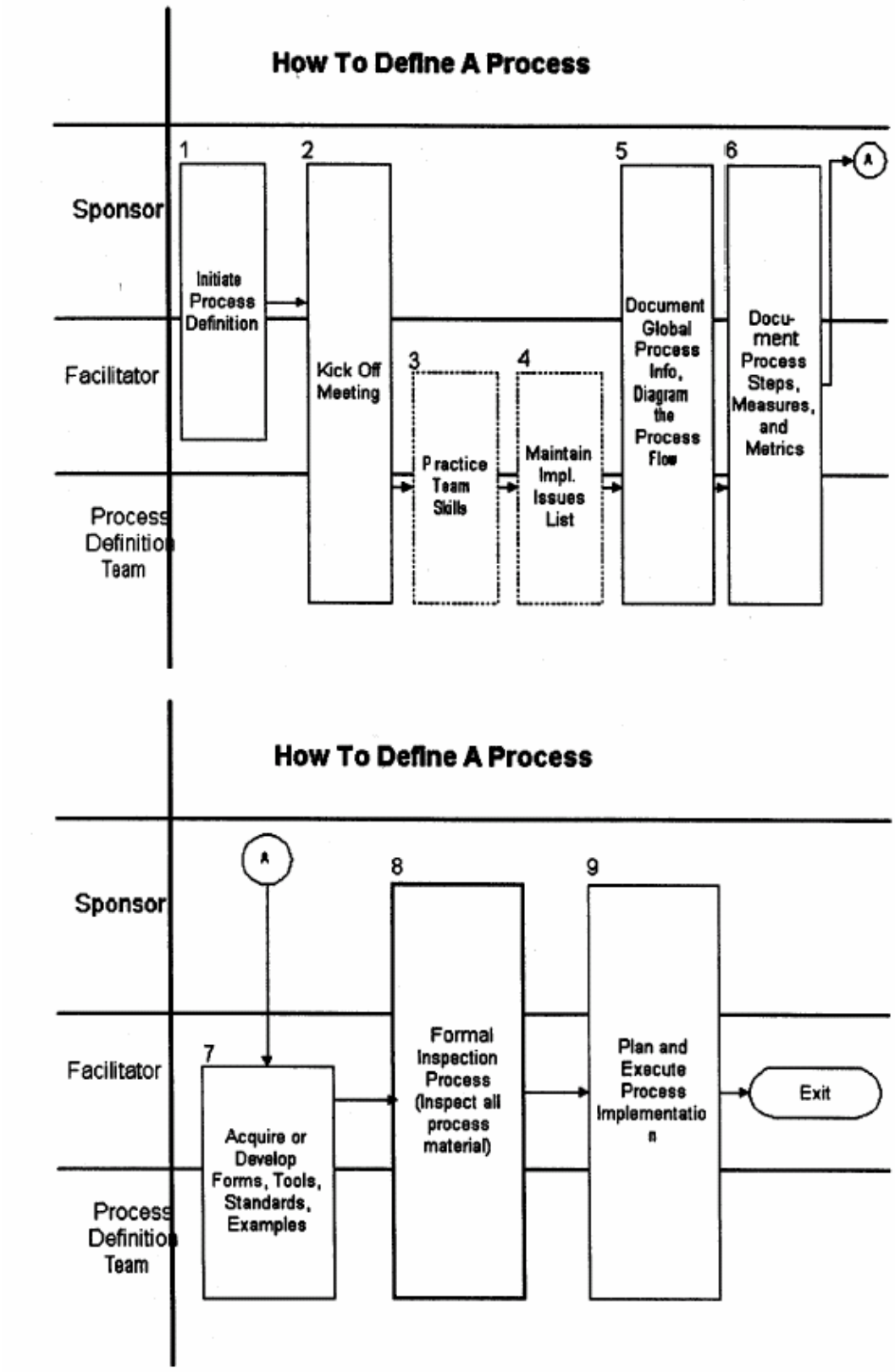

Figure 2: How to Define a Process

4. Current state of the process (e.g., what is working? What is not? Is there consistency?
   Redundancy? Extra Work?)
5. Process goals (What should this process accomplish? How will you know the process is a success? What should be measured? What will indicate a problem

with the process?)
6. Specific improvements to be addressed
7. Roles involved in the process
8. Primary internal and external customers of the process
9. Boundaries and constraints for the team (e.g., time allotted, target dates, equipment, costs, personnel, etc.)
10. Potential Process Definition Team members.

Step 2—Kick Off Meeting

The new Process Definition Team meets with the Sponsor and Facilitator to kick off the process work. The Sponsor and Facilitator have contacted people from the list of potential team members and now have a list of the team members. They now plan a meeting, provide the agenda, schedule the meeting, and lead the meeting.

The Kick Off Meeting is designed to:

1. Formalize involvement in the Process Definition Team,
2. Discuss the documented process goals and scope and verify that everyone is working with the same definitions,
3. Provide team members an overview of the process definition work, and,
4. Start the team working together right away by defining team rules and setting a meeting schedule.

Suggested Kick Off Meeting Agenda Items

- Process Sponsor - Introductions, purpose of meeting
- Process Sponsor - Presentation of process goals, scope
- Question and answer period
- Facilitator - Overview of process definition tasks
- Facilitator - Action Item assignment: who will handle the Action Item Log for the team? The Parking Lot log?[1]
- Facilitator - How often will the team meet? When? Where? Who will schedule the meetings and book the conference room?
- Facilitator - Work with team to decide on team name and team meeting rules
- Sponsor - Closing remarks

Step 3—Practice Team Skills

The Process Definition Team will be learning how to work together as a team in which each member contributes, and in which tools are used to make decisions and solve problems. The Facilitator needs to watch for opportunities to apply team-building exercises, problem-solving tools, and decision-making techniques. Tools that will be most used in process definition include brainstorming, negative brainstorming, affinity diagramming, selection matrix, cause/effect analysis, nominal group technique, data collection, Pareto Charts, force field analysis, plan-do- check-act cycle, and multivoting.

<u>Suggested Reference Material for the Facilitator</u>

The Team Handbook (How to Use Teams to Improve Quality) by Peter R. Scholtes and
The Quality Toolbox by Nancy R. Tague. Both are available from ASQ Quality Press.

Step 4—Maintain Implementation Issues List

During the definition of the process, implementation issues will be identified. The planning for implementation starts right away, with the creation of an Implementation Issues List. Topics are grouped and prioritized. Typical implementation issue types include training, tools, forms and templates. Implementation planning is an ongoing step until the process is completely documented. At that point, the Implementation Plan is formalized by complete documentation and inspection.

The Facilitator should be sure that the Process Definition Team identifies human factors that will affect implementation of the process. The team must plan ways to get non-team members involved with the process definition from the start; monthly presentations of the team's progress to their co-workers may be a good way to do this.

The implementation plan will have the most chance of success if the following are considered:

1. Everyone in the involved departments should understand that there must be a change to the way work is done, and why.
2. Everyone should understand the long term and short term goals, and how that will affect their department, their team, themselves.
3. Everyone should understand the steps that will be taken to meet those goals.
4. Most co-workers trust the Process Definition Team to be sure that before a change in work procedure is made, there will be training, tools, and examples provided.
5. Some people may require more coaching than others in new procedures. Everyone must understand that the Process Definition Team members will do everything they can to be sure that each individual can succeed in following the process to do their work.
6. It must be clear to everyone that the measurements and metrics defined for this process will never be used against an individual, but will be used for process improvement purposes.

Step 5—Document Global Process Information and Diagram the Process Flow

The Global Process Information

The Global Process Information is basically taken from Step 1—Initiate Process Definition, when the Sponsor provided information about the process scope and goals. At this point, the Process Definition Team will assign one individual member to be responsible for completing this documentation. The team can use the draft as a working copy, improving it until the inspection in Step 8.

The Contents of the Global Process Information Section

The Global Process Information is the first section of the Process Definition Document. It contains the following sections:

1. Overview (Basic description of the process.)
2. Objectives of the Process
3. Process Scope/Start and End
4. Parties Involved in the Process
5. Process Owner
6. Tools, Standards, Forms Used by this Process (A table listing name, use, and locations.)

7. Terms and Definitions
8. Related Documents (References to related documents, includes document name, description, and location. Include reference to the Process Metrics Plan.)

The Process Flow Diagram

The process flow diagramming is done in one or more team meetings, with the Facilitator leading. The recommended diagram type is that used in this paper, with an action zone for each role in the process.

Most teams' members know about the problems in their current, undefined way of doing work. In most cases, it is possible to diagram the way the team wants the improved process to be, without diagramming fully the way work is being done now. Usually the diagramming will be a combination of the two, with lots of changes. Regardless of how you get there, the end result should be a process flow diagram that is logical, contains all needed steps, does not show duplicated work, does not have work being done without purpose, and is easy to
follow.

The Process Flow Diagram: Instructions

If there are more than five roles in the process, group several together in one action zone, and as you draw the diagram just note which role is performing a step in that grouped zone. Usually in cases where there are more than five departments in a process, two or more of them have significantly fewer steps than other departments. These are ideal candidates for grouping on the diagram.

Line the meeting room walls with self-stick flip-chart paper. Draw lines to separate the action zones, write the role names in the appropriate zones, on the far left of the diagram. Now set the stage for the team by describing the state of the process right at the start. Ask what happens now. Draw process steps in the appropriate action zones; if a step will involve making a decision that affects process flow, put a decision point (diamond) symbol and write the decision question inside it. Keep asking, "what happens now?" until the last step in the process has been drawn.

Ask for a volunteer to copy the diagram and distribute copies before the next meeting.

At the next meeting, the team needs to walk through the process with as much realism as possible, playing roles, to find areas that need to be changed. Once the team agrees that the diagram will work, this step is done. Be sure the team understands that this is not set in stone—that as definition work progresses, the diagram will probably be changed.

The diagram will be put into a file using an online drawing tool, and incorporated into the Process Definition Document before Step 8—Inspection Process.

Step 6—Document Process Steps, Measures, and Metrics

First, each step is documented. The Process Definition Team members determine if one team member will document all the steps, or if the work will be split among members. For each step, the following information needs to be documented in the step description:
Purpose of the Step

1. Roles and Responsibilities (If more than one role is involved, who does what?)
2. Entry Criteria (How do you know you are supposed to be at this step?)

3. Inputs (What do you need in order to complete this step? Inputs include information and decisions.)
4. Next Step (Most of the time, the next step on the process flow diagram. If a decision point follows this step, document that here and tell where control will go based on the decision's value.)
5. Exit Criteria (How do you know you are really done with this step?)
6. Output (Outputs include decisions, information, communication, and products.)
7. Work Instructions (Exactly what to do, in order.)
8. Tools/Techniques (Are there any tools, forms, standards, or special techniques to use in following the work instructions? Where do I find them?)
9. Special Considerations (Are there exceptions to be considered for this step? Additional information you need the reader to know?)

Next, the team defines measures and metrics for the process, using the Goal/Question/Metric approach. A Process Metrics Plan is documented, which describes the overall metrics program and documents each selected metric in detail.

After metrics have been identified, documented, and selected, the step descriptions are reviewed to determine where measurement collection activities need to be in place, and these activities are added to the appropriate step descriptions.

The Facilitator needs to know metrics and measurement theory, and be well versed in the psychological factors that can derail a metrics program.

Recommended Metrics Planning Approach

"Managing Metrics Is Not an Oxymoron," Douglas B. Hamilton, published in the proceedings from the Seventh International Conference on Software Quality.

Step 7—Acquire or Develop Forms, Tools, Standards, Examples

The Process Definition Team examines each Step Description and determines if there is any need for a standard, tool, checklist, or template in completing the work described in the step. For example, for Step 5—Document Global Process Information, Diagram the Process Flow, two templates would make the work easier. We need a word processing template, "Process Definition Document" for documenting the global process information and a template in a graphics tool to use in documenting process flow diagrams.

The team needs to identify, design or specify, and plan to acquire or develop all items needed to implement the process. Usually a process can be implemented in phases, so the team needs to decide which tools must be available for the first implementation phase, and which can be purchased or developed later.

Step 8—Formal Inspection Process

The Formal Inspection process, designed to find defects in software deliverables, is also the best way to find defects in any documentation throughout the organization. The team uses the formal Inspection process to find defects in the process documentation—the Process Definition Document, the Process Metrics Plan, and any forms, templates, and standards developed for the process. The Facilitator is responsible for leading the Inspection effort, unless the organization already has an Inspection process in place with trained Inspection Leaders. This step is complete when all items have successfully passed Inspection.

Recommended Reference

Software Inspection, Tom Gilb and Dorothy Graham, is an excellent reference.

Step 9—Plan and Execute Process Implementation

At this point, the Process Definition Team should have a rather complete list of Implementation Issues on their list. A plan with a list of implementation and process management activities, estimated effort, and target dates is now documented and reviewed with the Sponsor. After the plan is approved, the team puts it into action.

1.8 The Managed Process

A managed process is a performed process that is planned and executed in accordance with policy; employs skilled people having adequate resources to produce controlled outputs; involves relevant stakeholders; is monitored, controlled, and reviewed; and is evaluated for adherence to its process description.

The process can be instantiated by a project, group, or organizational function. Management of the process is concerned with institutionalization and the achievement of other specific objectives established for the process, such as cost, schedule, and quality objectives. The control provided by a managed process helps to ensure that the established process is retained during times of stress.

The requirements and objectives for the process are established by the organization. The status of the work products and services are visible to management at defined points (e.g., at major milestones, on completion of major tasks). Commitments are established among those who perform the work and the relevant stakeholders and are revised as necessary. Work products are reviewed with relevant stakeholders and are controlled. The work products and services satisfy their specified requirements.

A critical distinction between a *performed process* and a *managed process* is the extent to which the process is managed. A managed process is planned (the plan can be part of a more encompassing plan) and the execution of the process is managed against the plan. Corrective actions are taken when the actual results and execution deviate significantly from the plan. A managed process achieves the objectives of the plan and is institutionalized for consistent execution.

1.9 The Optimization Process

The purpose of **optimization** is to achieve the "best" design relative to a set of prioritized criteria or constraints. These include maximizing factors such as productivity, strength, reliability, longevity, efficiency, and utilization. ... This decision-making process is known as **optimization**.

What is the purpose of optimization?

Optimization problem: Maximizing or minimizing some function relative to some set, often representing a range of choices available in a certain situation. The function allows comparison of the different choices for determining which might be "best." Common applications: Minimal cost, maximal profit, minimal error, optimal design, optimal management, variational principles.

Goals of the subject

The understanding of

Modeling issues—

What to look for in setting up an optimization problem?
What features are advantageous or disadvantageous?
What devices/tricks of formulation are available?
How can problems usefully be categorized?

Analysis of solutions—

What is meant by a "solution?"
When do solutions exist, and when are they unique?
How can solutions be recognized and characterized?
What happens to solutions under perturbations?

Numerical methods—

How can solutions be determined by iterative schemes of computation?
What modes of local simplification of a problem are convenient/ appropriate?
How can different solution techniques be compared and evaluated?

Distinguishing features of optimization as a mathematical discipline:

Descriptive −→ Prescriptive
Equations −→ Inequalities
Linear/Nonlinear −→ Convex/Nonconvex
Differential Calculus −→ Sub Differential Calculus

Finite-dimensional optimization:

The case where a choice corresponds to selecting the values of a finite number of real variables, called decision variables. For general purposes the decision variables may be denoted by x1, . . . , xn and each possible choice therefore identified with a point x = (x1, . . . , xn) in the space IRn . This is what we'll be focusing on in this course.

Feasible set: The subset C of IRn representing the allowable choices x = (x1, . . . , xn). Objective function: The function f0(x) = f0(x1, . . . , xn) that is to be maximized or minimized over C.

Constraints: Side conditions that are used to specify the feasible set C within IR^n .

Equality constraints: Conditions of the form $f_i(x) = c_i$ for certain functions f_i on IR^n and constants c_i in IR^n

Inequality constraints: Conditions of the form $fi(x) \leq ci$ or $fi(x) \geq ci$ for certain functions fi on IRn and constants ci in IR.

Range constraints: Conditions restricting the values of some decision variables to lie within certain closed intervals of IR. Very important in many situations, for instance, are

nonnegativity constraints: some variables xj may only be allowed to take values ≥ 0; the interval then is [0,∞). Range constraints can also arise from the desire to keep a variable between certain upper and lower bounds.

Linear constraints: Range constraints or conditions of the form fi(x) = ci , fi(x) ≤ ci , or fi(x) ≥ ci , in which the function is linear in the standard sense of being expressible as sum of constant coefficients times the variables x1, . . . , xn.

Data parameters: General problem statements usually involve not only decision variables but symbols designating known coefficients, constants, or other data elements. Conditions on such elements, such as the nonnegativity of a particular coefficient, are not among the "constraints" in a problem of optimization, since the numbers in question are supposed to be given and aren't subject to choice.

Mathematical programming: A traditional synonym for finite-dimensional optimization. This usage predates "computer programming," which actually arose from early attempts at solving optimization problems on computers. "Programming," with the meaning of optimization, survives in problem classifications such as linear programming, quadratic programming, convex programming, integer programming, etc.

1.10 Process Reference Models

1.11 Capability Maturity Model (CMM)

The Capability Maturity Model (CMM) is a methodology used to develop and refine an organization's software development process. The model describes a five-level evolutionary path of increasingly organized and systematically more mature processes. CMM was developed and is promoted by the Software Engineering Institute (SEI), a research and development center sponsored by the U.S. Department of Defense (DoD). SEI was founded in 1984 to address software engineering issues and, in a broad sense, to advance software engineering methodologies. More specifically, SEI was established to optimize the process of developing, acquiring, and maintaining heavily software-reliant systems for the DoD. Because the processes involved are equally applicable to the software industry as a whole, SEI advocates industry-wide adoption of the CMM.

The CMM is similar to ISO 9001, one of the ISO 9000 series of standards specified by the International Organization for Standardization (ISO). The ISO 9000 standards specify an effective quality system for manufacturing and service industries; ISO 9001 deals specifically with software development and maintenance. The main difference between the two systems lies in their respective purposes: ISO 9001 specifies a minimal acceptable quality level for software processes, while the CMM establishes a framework for continuous process improvement and is more explicit than the ISO standard in defining the means to be employed to that end.

Levels of CMM

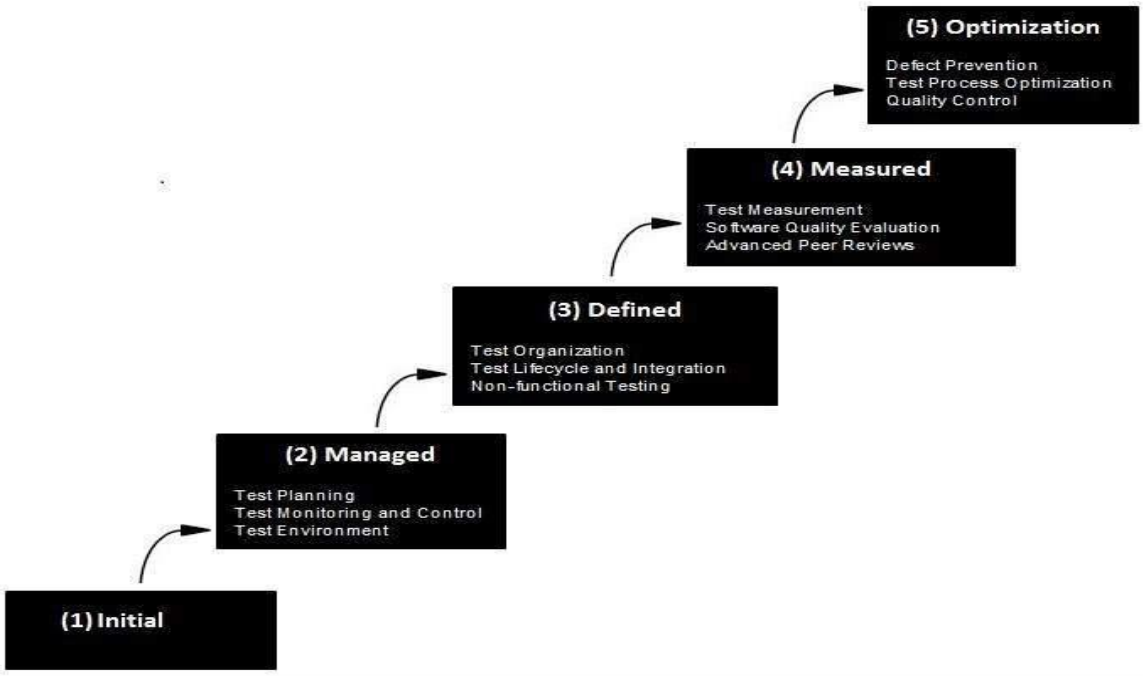

Figure 3: CMM Reference Model

- **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 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.

- **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.

1.12 Capability Maturity Model Integration

The Capability Maturity Model Integration (CMMI) is a process and behavioral model that helps organizations streamline process improvement and encourage productive, efficient behaviors that decrease risks in software, product and service development.

The CMMI was developed by the Software Engineering Institute at Carnegie Mellon University as a process improvement tool for projects, divisions or organizations. The DoD and U.S. Government helped develop the CMMI, which is a common requirement for DoD and U.S. Government software development contracts. The CMMI is currently administered by the CMMI Institute, which was purchased by the ISACA in 2016.

CMMI model

The CMMI starts with an appraisal process that evaluates three specific areas: process and service development, service establishment and management, and product and service acquisition. It's designed to help improve performance by providing businesses with everything they need to consistently develop better products and services.

But the CMMI is more than a process model; it's also a behavioral model. Businesses can use the CMMI to tackle the logistics of improving performance by developing measurable benchmarks, but it can also create a structure for encouraging productive, efficient behavior throughout the organization.

Standard CMMI Appraisal Method For Process Improvement (Scampi)

SCAMPI is the official appraisal method used by the CMMI institute. It is outlined in the SCAMPI Method Definition Document, which is included in the CMMI appraisal reference documents. There are three appraisal classes: Class A, B and C.

Scampi A

The most rigorous appraisal method, SCAMPI A is most useful after multiple processes have been implemented. It provides a benchmark for businesses and is the only level that results in an official rating. It must be performed by a certified lead appraiser, who is part of the on-site appraisal team.

Scampi B

This appraisal is less formal than SCAMPI A; it helps find a target CMMI maturity level, predict success for evaluated practices and give the business a better idea of where they stand in the maturity process.

Scampi C

This appraisal method is shorter, more flexible and less expensive than Class A or B. It's designed to quickly assess a business's established practices and how those will integrate or align with CMMI practices. It can be used at a high-level or micro-level, to address organizational issues or smaller process or departmental issues. It involves more risk than Class A or B, but it's more cost-effective.

CMMI Maturity Levels

The CMMI model breaks down organizational maturity into five levels. For businesses that embrace CMMI, the goal is to raise the organization up to Level 5, the "optimizing" maturity level. Once businesses reach this level, they aren't done with the CMMI. Instead, they focus on maintenance and regular improvements.

CMMI's five Maturity Levels are:
Initial

Processes are viewed as unpredictable and reactive. At this stage, "work gets completed but it's often delayed and over budget." This is the worst stage a business can find itself in — an unpredictable environment that increases risk and inefficiency.

Managed

There's a level of project management achieved. Projects are "planned, performed, measured and controlled" at this level, but there are still a lot of issues to address.

Defined

At this stage, organizations are more proactive than reactive. There's a set of "organization-wide standards" to "provide guidance across projects, programs and portfolios." Businesses understand their shortcomings, how to address them and what the goal is for improvement.

Quantitatively managed

This stage is more measured and controlled. The organization is working off quantitative data to determine predictable processes that align with stakeholder needs. The business is ahead of risks, with more data-driven insight into process deficiencies.
Optimizing: Here, an organization's processes are stable and flexible. At this final stage, an organization will be in constant state of improving and responding to changes or other opportunities. The organization is stable, which allows for more "agility and innovation," in a predictable environment.

Once organizations hit Levels 4 and 5, they are considered high maturity, where they are "continuously evolving, adapting and growing to meet the needs of stakeholders and customers." That is the goal of the CMMI: To create reliable environments, where products, services and departments are proactive, efficient and productive.

Maturity Levels And Process Areas

Here is a list of all the corresponding process areas defined for a S/W organization. These process areas may be different for different organization.

This section is just giving names of the related process areas, for more detail about these Process Areas go through CMMI Process Areas Chapter.

Table 1: Maturity Level and Process Areas

Level	Focus	Key Process Area	Result
5 Optimizing	Continuous Process Improvement	• Organizational Innovation and Deployment • Causal Analysis and Resolution	Highest Quality / Lowest Risk
4 Quantitatively Managed	Quantitatively Managed	• Organizational Process Performance • Quantitative Project Management	Higher Quality / Lower Risk
3 Defined	Process Standardization	• Requirements Development • Technical Solution • Product Integration • Verification • Validation • Organizational Process Focus • Organizational Process Definition • Organizational Training • Integrated Project Mgmt (with IPPD extras) • Risk Management • Decision Analysis and Resolution • Integrated Teaming (IPPD only) • Org. Environment for Integration (IPPD only) • Integrated Supplier Management (SS only)	Medium Quality / Medium Risk

2 Managed	Basic Project Management	• Requirements Management • Project Planning • Project Monitoring and Control • Supplier Agreement Management • Measurement and Analysis • Process and Product Quality Assurance • Configuration Management	Low Quality / High Risk
1 Initial	Process is informal and Adhoc		Lowest Quality / Highest Risk

1.13 People Capability Maturity Model (PCMM)

People Capability Maturity Model (PCMM) is a maturity framework that focuses on continuously improving the management and development of the human assets of a software or information systems organization. PCMM can be perceived as the application of the principles of Capability Maturity Model to human assets of a software organization. It describes an evolutionary improvement path from ad hoc, inconsistently performed practices, to a mature, disciplined, and continuously improving development of the knowledge, skills, and motivation of the workforce. Although the focus in People CMM is on software or information system organizations, the processes and practices are applicable for any organization that aims to improve the capability of its workforce. PCMM will be guiding and effective particularly for organizations whose core processes are knowledge intensive.

The primary objective of the People Capability Maturity Model is to improve the capability of the entire workforce. This can be defined as the level of knowledge, skills, and

process abilities available for performing an organization's current and future business activities.

10 Principles of People Capability Maturity Model (PCMM)

The People Capability Maturity Model describes an evolutionary improvement path from ad hoc, inconsistently performed workforce practices, to a mature infrastructure of practices for continuously elevating workforce capability. The philosophy implicit the PCMM can be summarized in ten principles.

1. In mature organizations, workforce capability is directly related to business performance.
2. Workforce capability is a competitive issue and a source of strategic advantage.
3. Workforce capability must be defined in relation to the organization's strategic business objectives.
4. Knowledge-intense work shifts the focus from job elements to workforce competencies.
5. Capability can be measured and improved at multiple levels, including individuals, workgroups, workforce competencies, and the organization.
6. An organization should invest in improving the capability of those workforce competencies that are critical to its core competency as a business.
7. Operational management is responsible for the capability of the workforce.
8. The improvement of workforce capability can be pursued as a process composed from proven practices and procedures.
9. The organization is responsible for providing improvement opportunities, while individuals are responsible for taking advantage of them.
10. Since technologies and organizational forms evolve rapidly, organizations must continually evolve their workforce practices and develop new workforce competencies.

The People Capability Maturity Model (People CMM) is a roadmap for implementing workforce practices that continuously improve the capability of an organization's workforce. Since an organization cannot implement all of the best workforce practices in an afternoon, the People CMM introduces them in stages. Each progressive level of the People CMM produces a unique transformation in the organization's culture by equipping it with more powerful practices for attracting, developing, organizing, motivating, and retaining its workforce. Thus, the People CMM establishes an integrated system of workforce practices that matures through increasing alignment with the organization's business objectives, performance, and changing needs.

Although the People CMM has been designed primarily for application in knowledge intense organizations, with appropriate tailoring it can be applied in almost any organizational setting. The People CMM's primary objective is to improve the capability of the workforce. Workforce capability can be defined as the level of knowledge, skills, and process abilities available for performing an organization's business activities.

People Capability Maturity Model (PCMM) Levels

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:

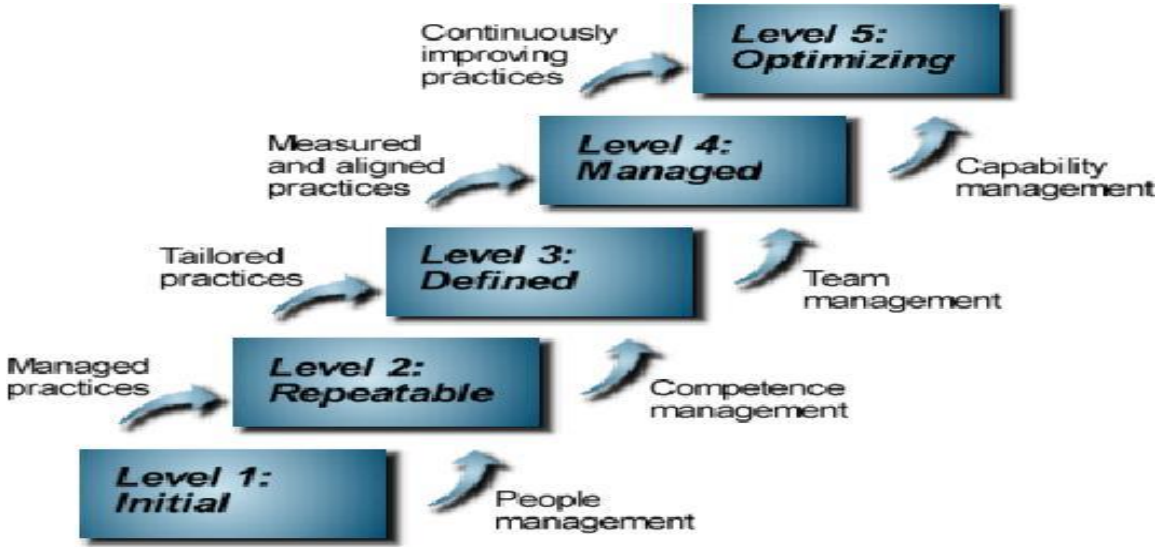

Figure 4: PCCM Levels

Table 2: PCMM Threads

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 Work Environment	Staffing

1.14 Personal Software Process

The PSP is a personal process for developing software or for doing any other defined activity. The PSP includes
defined steps
forms
standards

It provides a measurement and analysis framework for characterizing and managing your personal work.

It is also a defined procedure that helps you to improve your personal performance.

PSP Process Flow

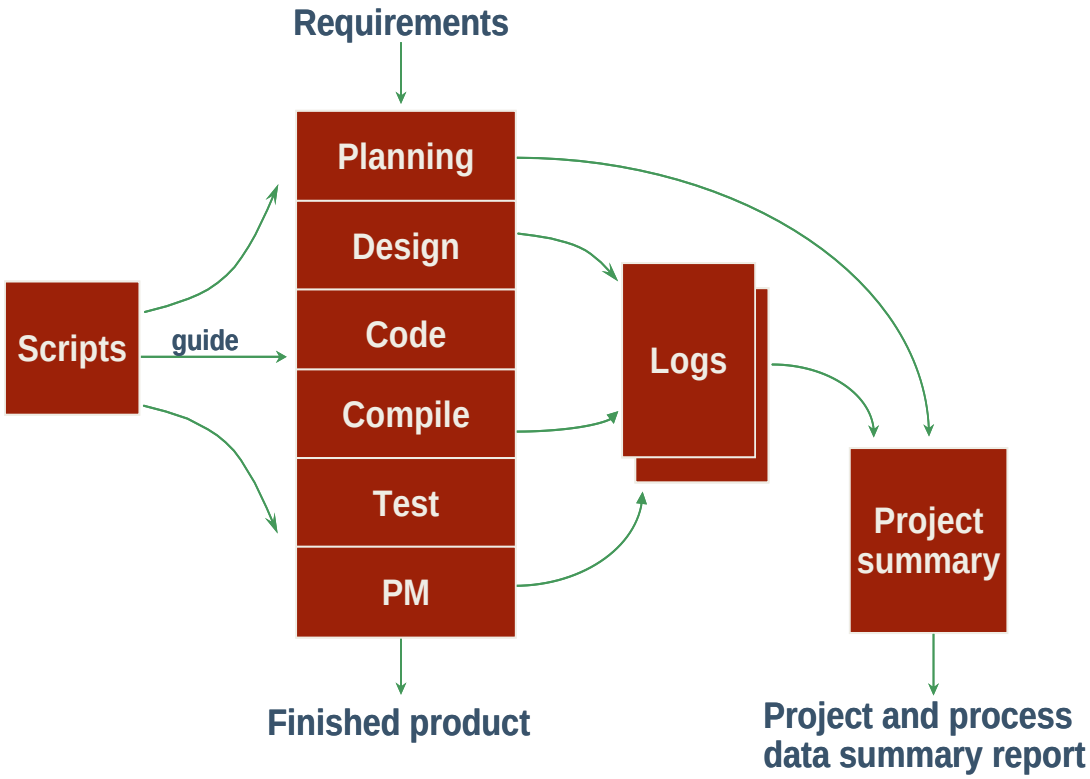

Figure 5: PSP Process Flow Diagram

The PSP process is designed for individual use.
It is based on scaled-down industrial software practice.
The PSP course demonstrates the value of using a defined and measured process.
It helps you and your organization meet the increasing demands for high quality and timely software.

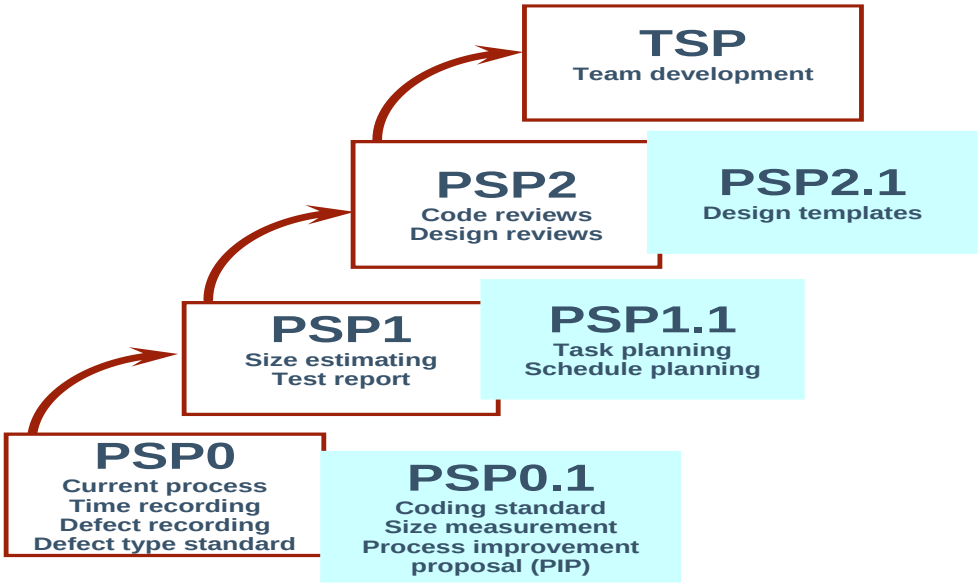

Figure 6: PSP Phases Diagram

Table 3: PSP Phases

Phase	Emphasis	Features
0	Personal Management	Current process plus basic measures: development time, defects injected and removed; process: planning, development, analysis
0.1		Coding standards, process improvement proposal form, size measurements
1	Personal Planning	PROBE; Size estimation, time estimates, test report
1.1		Task planning, schedule planning
2	Personal Quality	Defect management: code reviews, design reviews
2.1		Design specification and analysis; defect prevention; process analysis; process benchmarks
3	Scaling Up	Cyclic development

Learning PSP 0

- Personal measurement
- Forms and scripts
- Time, defects injected and removed
- Phases: planning, development, postmortem
- PSP0.1: add in coding standards, size measurement, and process improvement proposal

Learning PSP 1

- Personal planning
- PROBE estimation; confidence intervals
- PSP1.1: schedule and task planning

Table 4: PSP1 Process Script (SEI)

Phase Number	Purpose	To guide you in developing module-level programs
	Entry Criteria	• Problem description • PSP1 Project Plan Summary form • ***Size Estimating Template*** • ***Historical estimate and actual size data*** • Time and Defect Recording Logs • Defect Type Standard • Stop watch (optional)
1	Planning	• Produce or obtain a requirements statement. • ***Use the PROBE method to*** estimate the total new and changed LOC required. • ***Complete the Size Estimate Template.*** • Estimate the required development time. • Enter the plan data in the Project Plan Summary form. • Complete the Time Recording Log.
2	Development	• Design the program. • Implement the design. • Compile the program and fix and log all defects found. • Test the program and fix and log all defects found. • Complete the Time Recording Log.
3	Postmortem	Complete the Project Plan Summary form with actual time, defect, and size data.
	Exit Criteria	• A thoroughly tested program • Completed Project Plan Summary form with estimated and actual data • ***Completed Size Estimating Template*** • ***Completed Test Report Template*** • Completed PIP forms • Completed Defect and Time Recording Logs

Learn PSP 2

- Personal quality
- Defect management: data, review checklists
- PSP2.1: design specification, defect prevention, process analysis, process benchmarks
-

Learn PSP 3:

- Scaling up
- Cyclic development
- Design verification; process definition principles
- Subsumed by TSP

Overall PSP Strategy

1. Gather data
2. Estimate and plan
3. Manage defects
4. Manage yield
5. Control cost of quality

Gather Data

- **Measurements taken**
  - Time in each process activity (and for interrupts)
  - Defects introduced and removed for each activity
  - Developed product size (LOC)
    - Base, added, modified, deleted, new and changed, reused, new reuse, total
- **Metrics computed**
  - Size and time estimating error
  - Cost-performance index
  - Defect
    - Injected and removed per hour
    - Density
  - Process yield
  - Appraisal and failure cost of quality
  - Appraisal to failure ratio
  -

Estimate and Plan

PROBE - proxy based estimation method
PSP proxies: functions and object
Others include function points, screens, reports, sections of text
Linear regression on at least 3 prior projects
Goal is to improve estimates over time
PSP students improved their size estimates from 31% (within 20%) to 42% between programs one and ten
Improved time estimates from 33% (within 20%) to 49%

SEI

- **Base** When an existing product is enhanced, base LOC is the size of the original product version before any modifications are made.
- **Added** Code written for a new program or added to an existing base program.
- **Modified** LOC in an existing (Base) program that are changed.
- **Deleted** LOC in an existing (Base) program that are deleted.
- **New and Changed** When engineers develop software, it takes them much more time to add or modify a LOC than it does to delete or reuse one. Thus, in the PSP, engineers use only the **Added** or **Modified** code to make size and resource estimates. This code is called the **New and Changed** LOC.
- **Reused** Code taken from a reuse library and used, without modification, in a new program. Reuse does not count the unmodified base code retained from a prior program version and it does not count any code that is reused with modifications.
- **New Reuse** LOC that an engineer develops and contributes to the reuse library.
- **Total** Total size of a program, regardless of its source (= **Base - Deleted** + **Added** + **Reuse**).
- Record, for each defect
  - Activity (phase) during which defect was injected and removed
    - Planning, design, design review, code, code review, compile, test

  - Defect type (next slide)
  - Fix time
  - Description
- Students reduced defect rates from 116/KLOC to 49/KLOC between programs one and ten
  - Standard deviation also reduced

Table 5: Defect Types

Type Number	Type Name	Description
10	Documentation	comments, messages
20	Syntax	spelling, punctuation, types, instruction formats
30	Build, package	change management, library, version control
40	Assignment	declaration, duplicate name, scope, limits
50	Interface	procedure calls and references, I/O, user format
60	Checking	error messages, inadequate checks
70	Data	structure, content
80	Function	logic, pointers, loops, recursion, computations, function defects
90	System	configuration, timing, memory
100	Environment	design, compile, test, or other support-system problems

Manage Yield

- Yield is PSP's principle quality measure
- If it is costly to find a defect during testing, then you need to find it earlier (during review)
  - (Or not insert it in the first place)
- Hold review before compilation
  - (But aren't compilers cheaper than programmers?)
  - (And desk check every new compilation)
- *Yield:* % defects found and fixed before compilation
  - Engineers review code before first compile
  - 9% of "syntax" error get by compiler
  - Defects found at compile time correlate with defects found during test (r = .71)
  - Strong correlation between defects found during test and customer failures (r = .91)
- Introduction of design and code reviews strongly improves yield

Control Cost of Quality

- Appraisal cost
  - Time spent in design and code reviews
- Failure cost
  - Time spent in compile and test
- Prevention costs
  - Prototyping, formal specification
  - Not part of PSP
- Appraisal to failure ratio (A/FR)
  - Raise until quality is sufficient then gradually lower
  - Initial target at least two

PSP Benefits

- Increases personal commitment by investing each engineer with process responsibility
- Assists engineers in making accurate plans
- Provides steps engineers can take to improve personal and project quality
- Sets benchmarks to measure personal process improvements
- Demonstrates the impact of process changes on an engineer's performance

Team Software Process (Tsp)

In combination with the personal software process (PSP), 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.

TSP Strategy

- Provide a simple process framework based on the PSP.
- Use modest, well-defined problems.
- Develop products in several cycles.
- Establish standard measures for quality and performance.
- Provide detailed role definitions.
- Use role and team evaluations.
- Require process discipline.
- Provide guidance on teamwork problems.

1.15 TSP Process

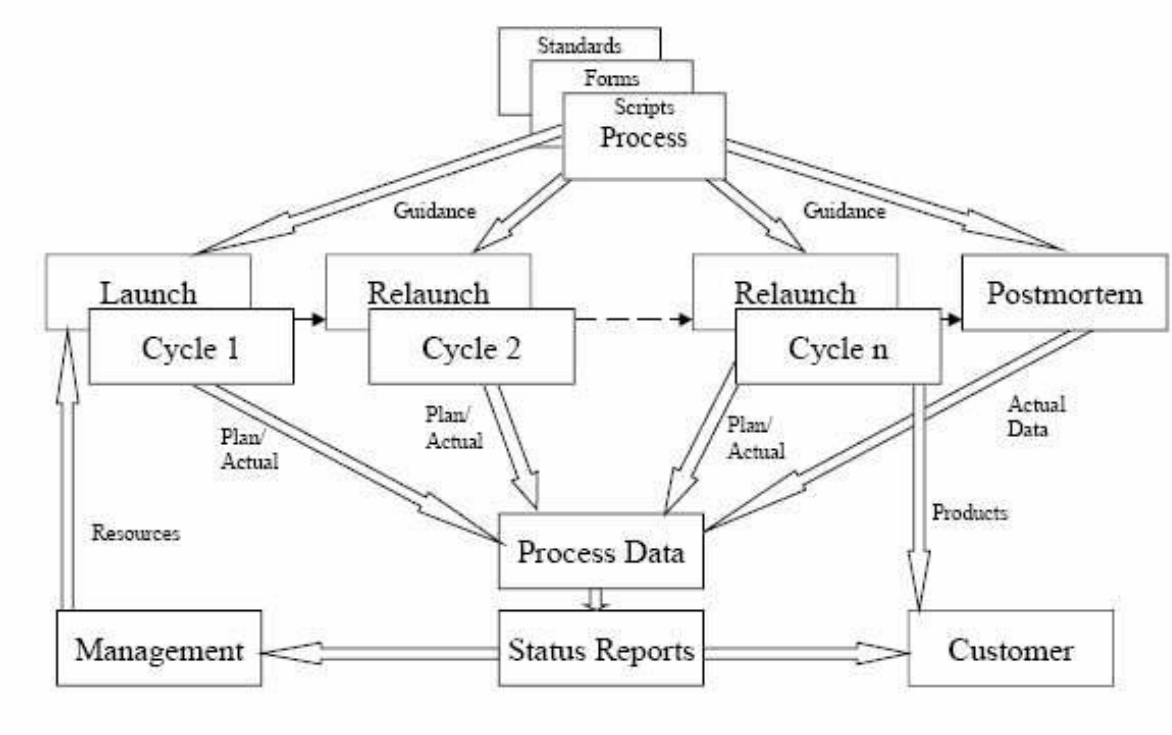

Figure 7: TSP Process Flow

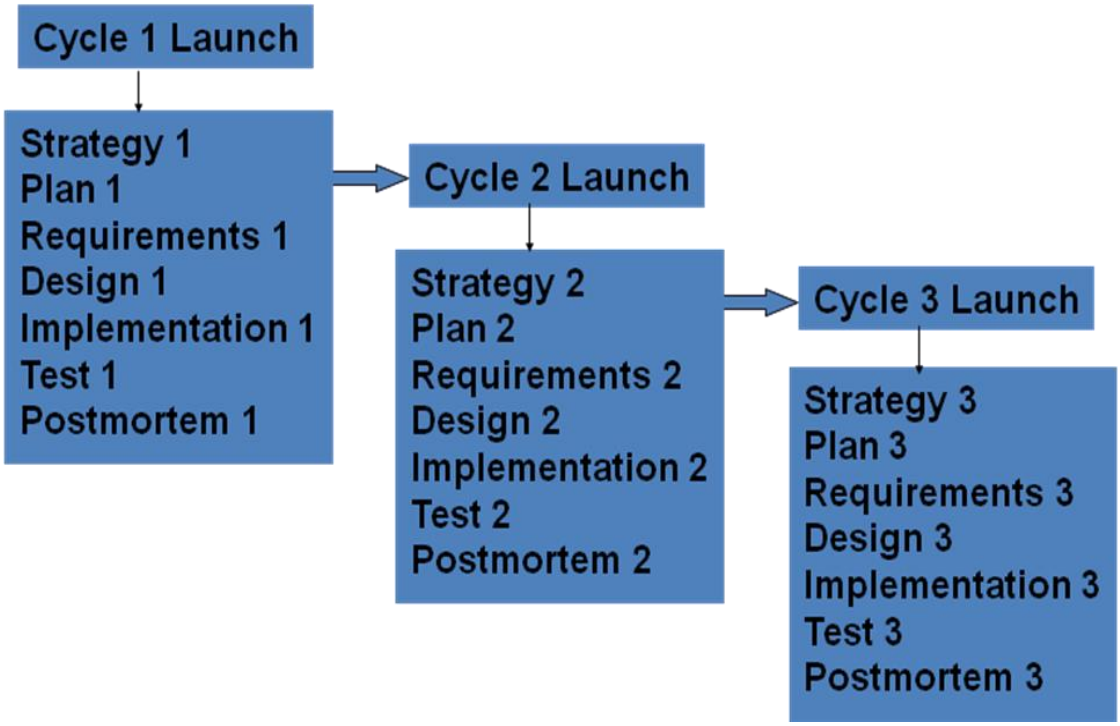

Figure 8: TSP Cycle Diagram

Life Cycle Phases

- Launch
- Strategy
- Plan
- Requirements
- Design
- Implement
- Test
- Postmortem

- **Launch**

  - review course objectives
  - describe the TSP structure and content
  - assign student teams and roles
  - describe the customer needs statement
  - establish team and individual goals

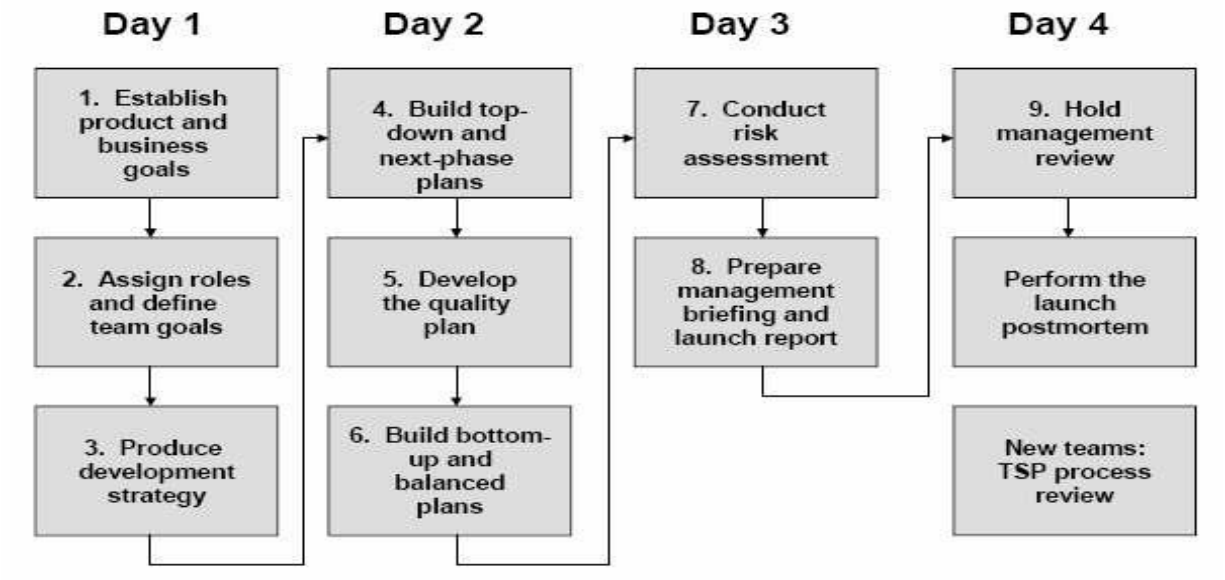

Figure 9: Launch Process

- **Strategy**

  - create a conceptual design for the product
  - establish the development strategy: decide what will be produced in each cycle
  - make initial size and effort estimates
  - establish a configuration management plan
  - Reuse plan
  - Risk Management

- **Planning**

  - estimate the size of each artifact to be developed (SRS, SDS, code)

- identify tasks to be performed; estimate time to complete each task; assign tasks to team members
- make a weekly schedule for task completion
- make a quality plan

- **Requirements**

  - analyze need statement and interview customer
  - specify the requirements
  - inspect the requirements
  - develop a system test plan

- **Design**

  - create a high-level design
  - specify the design
  - inspect the design
  - develop an integration test plan

- **Implementation**

  - use the PSP to implement modules/units
  - create detailed design of modules/units
  - review the design
  - translate the design to code
  - review the code
  - compile and test the modules/units
  - analyze the quality of the modules/units

- **Test**

  - build and integrate the system
  - conduct a system test
  - produce user documentation

- **Postmortem**

  - conduct a postmortem analysis
  - write a cycle report
  - produce peer and team evaluations

Team (Dyer)

A team consists of at least two people.

- The members are working toward a common goal.
- Each person has a specific assigned role.
- Completion of the mission requires some form of dependency among the group members.

Effective Teams

- The members are skilled.
- The team's goal is important, defined, visible, and realistic.
- The team's resources are adequate for the job.
- The members are motivated and committed to meeting the team's goal.
- The members cooperate and support each other.
- The members are disciplined in their work.

Team Building

- The team members establish common goals and defined roles.
- The team develops an agreed-upon strategy.
- The team members define a common process for their work.
- All team members participate in producing the plan, and each member knows his or her personal role in that plan.
- The team negotiates the plan with management.
- Management reviews and accepts the negotiated plan.
- The team members do the job in the way that they have planned to do it.
- The team members communicate freely and often.
- The team forms a cohesive group: the members cooperate, and they are all committed to meeting the goal.
- The engineers know their status, get feedback on their work, and have leadership that sustains their motivation.

Selecting Roles

- Team Leader
- Development Manager
- Planning Manager
- Quality/Process Manager
- Support Manager
- Customer interface manager
- Design manager
- Test manager
- Safety manager
- Security manager
- Performance manager

Team Leader Responsibilities

- Motivating team members
- Handling customer issues
- Interaction with management
- Day-to-day direction of the work
- Protecting team resources
- Resolving team issues
- Conducting team meetings
- Reporting on the work status

Development Manager

• Leads and guides the team in designing and developing the product
  – Lead the team in producing the development strategy and the product conceptual design
  – Lead the team in producing the design specification (SDS)
• If there is no separate Design Manager or Software Architect
  – Lead the team in implementing the product

Planning Manager

• Supports and guides the team in planning and tracking their work
  – Lead the team in producing the task plan and schedule for each development cycle
  – Lead the team in producing the balanced team development plan
  – Track the team's progress against their plan

Quality / Process Manager

- Supports the team in defining their process needs, in making the quality plan and in tracking process and product quality
  – Lead the team in producing and tracking their quality plan
  – Identify where quality performance falls short of objectives.
  – Lead the team in defining, documenting, and maintaining their processes and development standards
  – Act as moderator and lead all team reviews and inspections

Support Manager

• Supports the team in determining, obtaining, and managing the tools needed to meet its technology and administrative support needs
  – Lead the team in determining their support needs and obtaining the needed tools and facilities
  – Lead the development and management of Change/Configuration Management System
  – Handle the team's issue and risk tracking system
  – Act as the team's reuse advocate

Task Planning

• Generate a default task list
• Modify the default list as appropriate and estimate sizes and times for each task
• Assign portions of tasks to individual engineers
• Determine total time required for the project cycle

Schedule Planning

• Use LOC estimates and LOC/hour rates for time estimation
• Generate TASK and SCHEDULE plans
  – Estimate times based upon LOC and LOC/hour estimates
  – Estimate time available for the team

Produce Size Estimates

- Produce the conceptual design
  - Objects for all cycles
- Select a development strategy
  - Allocate objects to cycles
  - Produce a minimal working subset in the first cycle
  - Development Manager leads

- Produce preliminary size estimates
  - Estimate LOC in each class
  - Use size estimates as a basis for allocation of tasks to cycles
  - Enter estimates on the STRAT form

Quality Planning

- Enter default quality criteria into the SUMQ form
  - the Percent Defect Free components in each of the defect detection phases
  - the defect removal yields in various phases (Phase Yields)
  - the rate at which defects are injected in various phases (Defect Injection Rates)

TSP Quality Guidelines

- Percent (of modules) Defect Free (PDF) at entrance to
  - Compile > 10%
  - Unit Test > 50%
  - Integration Test > 70%
  - System Test > 90%

- Defects/KLOC:
  - Total defects injected 75 - 150; If not PSP trained, use 100 to 200.
  - Compile < 10
  - Unit Test < 5
  - Integration Test < 0.5
  - System Test < 0.2

- Defect Ratios
  - Detailed design review defects /unit test defects > 2.0
  - Code review defects/compile defects > 2.0

The Yield Measure

Phase yield is the percentage of defects entering and injected in a phase that are removed in that phase

Process yield is the yield of all phases up to that point in the process

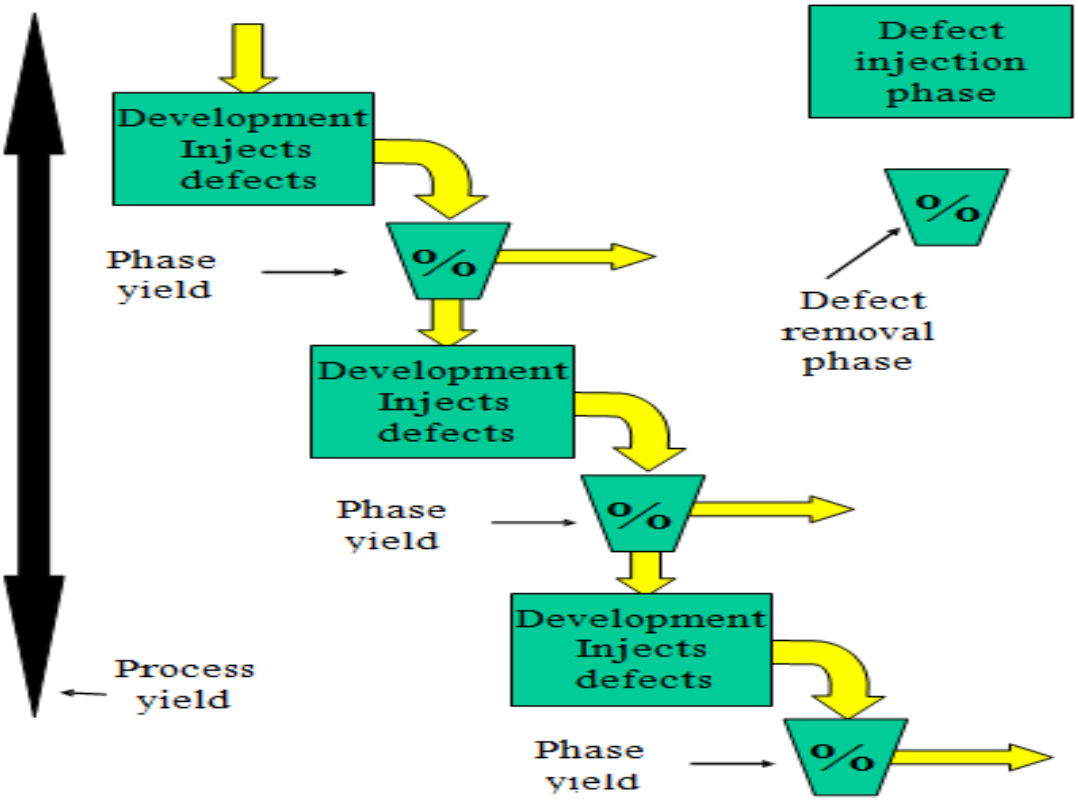

Figure 10: Yield Process

Phase Yields

- Team requirements inspections ~ 70%
- Design reviews and inspections ~ 70%
- Code reviews and inspections ~ 70%
- Compiling ~ 50% (90+ % of syntax defects)
- Unit test - at 5 or less defects/KLOC ~ 90%
  - For high defects/KLOC - 50-75%
- Integration and system test - at < 1.0 defects/KLOC ~ 80%
  - For high defects/KLOC - 30-65%
- Before compile >75%
- Before unit test > 85%
- Before integration test > 97.5%
- Before system test > 99%