SPIMS: Software Process Improvement Model for Small Scale Industries

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

By:
Dinesh Tagra
(80731027)

Under the supervision of:
Ms. Ashima Singh
Lecturer

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2009

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"SPIMS: Software Process Improvement Model for Small Scale Industries"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering and submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Ms. Ashima Singh and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Dinesh Tagra)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

June 16, 2009

(Ashima Singh)
Lecturer
Computer Science and Engineering Department,
Thapar University, Patiala.

Countersigned by:

18/06/2009

(SEEMA BAWA)
Professor & Head,
Computer Science & Engineering Department,
Thapar University,
Patiala.

9/7/09

(R.K.SHARMA)
Dean (Academic Affairs),
Thapar University,
Patiala.

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my guide **Ms. Ashima Singh,** Lecturer, Computer Science and Engineering Department for immense help, guidance, stimulating suggestions, and encouragement all the time with this thesis work. This work would have not been possible without her encouragement. She always provided a motivating and enthusiastic atmosphere to work with. It was a great pleasure to do this thesis under her supervision.

I am equally grateful to **Ms. Seema Bawa,** Professor and Head, Computer Science and Engineering Department for their appreciation and satisfactorily healing me off my inexperienced inquisitions about the new subject.

I am greatful to **Dr. R.K. Sharma**, Dean of Academic Affair for his constant encouragement that was of great importance in the completion of the thesis

I would also like to thank all the staff members and PhD scholar Ms. Shashi Bhanwar who was always there at the need of the hours and provided with all help and facilities, which I required for the completion of my thesis. I am deeply indebted to my parents and friends for their inspiration and ever encouraging moral support, which enabled me to pursue my studies.

I am also very thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct or indirect help, cooperation, love and affection which made my stay at Thapar University memorable.

Dinesh Tagra

(80731027)

Abstract

Software Process Improvement starts when organization starts thinking about the change in working environment so that performance and profit can be increased. The organizations with fewer resources needs better process improvement as mostly software are developed and supported by the small sized organizations. The improvements can not only be seen in processes but also in the final product. Hence, a small size organization can dream to become a large organization in future by making name, having more profit and high quality products.

In the present scenario in small scale industries, no standard process improvement model is used. SPICE model is meant for large, medium and small organization. BOOTSTRAP model is used for small and medium organization and PSP for small organization. But these models are not very popular among small scale organization because of cost, time and limited resources tradeoffs. They continue to engineer the software in a random and unorganized way which leads to feeble quality products.

SPIMS is a phased as well as an iterative model which is successful in establishing specialized cum generalized set of processes at the discretion or will of small scale industries. Small organizations are not capable of bearing the cost of establishing software process improvement programs and the existing software process standards or models are not easily applicable. Thus, a SPIMS model is proposed. SPIMS is an iterative model because it enhances and develops a software system incrementally. SPIMS is a specialized model because it incorporates some specialized techniques introduced in different phases. These new introduced specialized techniques in help to reduce cost, time and also improve product quality simultaneously. SPIMS also have generalized processes in supporting set which can be implemented effectively. SPIMS is capable of delivery the quality products which can withstand in market competition.

Table of Contents

List of Figures

`List of Tables

Chapter 1
Introduction

1.1 Software Process Improvement

Software Process Improvement is the name given to the identification of the current state of the practice of information systems development within an organization and then improving it. Software Process Improvement is an important activity which starts when an organization plans to enhance the capabilities of its ongoing processes. The principle objective of a mature software process is to produce quality products to meet customers' needs. SPI is used to improve organizational capabilities to deliver quality software by defined processes or systematic procedures. Software process improvement always facilitates to identify and apply the changes to current processes so that the new processes can be help in producing the high quality product. SPI is a difficult activity to initiate because of its complex and dynamic nature of processes. One major characteristic of process improvement is to emphasize the continuous improvement of products as well as of organizational processes in terms of performance, stability, compatibility [1].

1.2 Software Processes

Software process is a set of activities that begin with the identification of a need and concludes with the retirement of a product that satisfies the need or more completely as a set of activities, methods, practices, and transformations that people use to develop and maintain software and its associated products (e.g., project plans, design documents, code, test cases, user manuals) [2].

A structured set of activities required to develop a software system specification, design and implementation, validation and evolution. Software process capability describes the range of expected results achieved from a software process. But capability is not the same as performance. Software process performance is the actual results achieved from following a software process. That is, results achieved (performance) differ from results expected (capability).

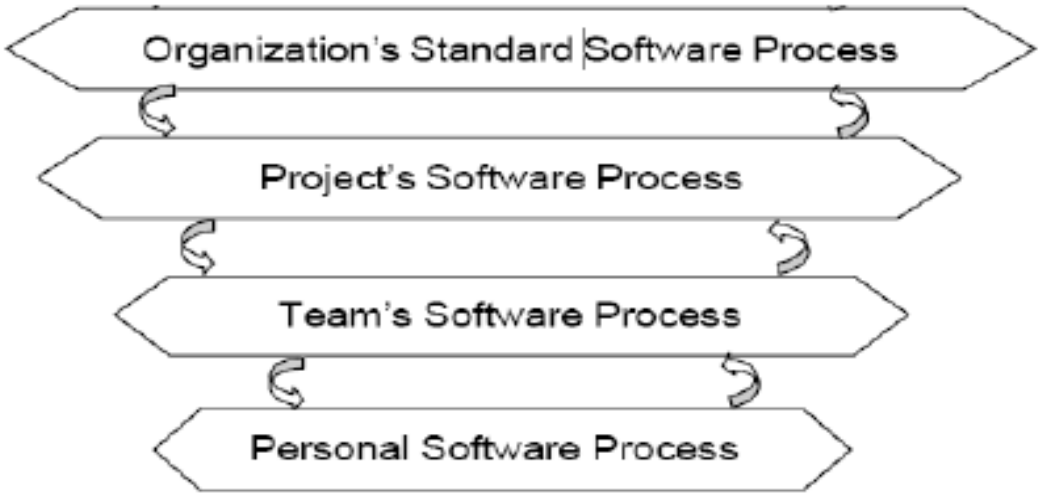

Figure 1: Software Processes [2]

1.2.1 Individual Processes

This is the basic level of the processes. These processes start as individual process and start moving towards the upper level. It is the foundational level to initiate a process. It is very important to have the good individual processes because these processes become the foundation of the higher level processes [2]. These processes are collaborated with the individuals.

1.2.2 Team Processes

These processes are formed and maintained at team levels where a number of people work together to combine the individual processes. Team processes also contain roles and responsibilities [2].

1.2.3 Project Processes

These processes are formed when a project is started having some specific targets to achieve. Processes involved at this level vary from project to project.

1.2.4 Business Processes

These are the highest level of processes containing the business visions, business strategies, business objectives and orientation which started from different individual process and end at this level [2].

1.2.5 Personal Software Processes

These processes can affect the overall progress and effectiveness of the organization. Improvement and effectiveness in the individual processes can lead towards the overall and aggregate improvement in the organizational performance [2] [3].

1.3 Basic Steps for Process Improvement

The opportunity for improvement to either operating or management processes can often be vast, but must be focused. It is imperative that the number of process improvement activities undertaken by an organization is matched by the organization's ability to fund the activity and implement the changes without harmful disruption to day-to-day delivery of its products and services [4].

The six basic steps for Process Improvement are [5]

- Process selection
- Process understanding
- Process performance
- Process review
- Process change
- Capturing the change

The objective of **Process Selection** is to select a small and achievable number of processes, most directly influencing the achievement of the organization's goals and objectives, upon which to undertake process improvement activity. This can take anywhere from a few hours to weeks, be either proactive, e.g. management initiative, or reactive, e.g., customer complaint, and involve one or several people.

The outcomes of the **Process Selection** step should be an agreed number of processes to be reviewed, management approval to dedicate resource to the work and agreed objectives for the work.

Next, comes **Process Understanding**, covering the scope of the process – where it starts and ends, what is included and excluded. In addition, the key sub-processes and accountabilities of the process to the organization must be understood. These can be achieved by completing the elements of a process - title, purpose, scope, inputs, outputs, controls and resources, and using tools such as process mapping and decomposition.

The outcomes of **Process Understanding** are a high level process map, sub-process maps, a list of key accountabilities and lists of the major inputs, outputs, controls and resources acting upon the processes and sub-processes.

Process Performance involves recording and detailing the historical performance of the process, obtaining perceptual views of both current and historical performance from customers and suppliers, defining the agreed required performance of the future improved process, and agreeing how it will be measured, monitored and reviewed. Data must be gathered and analysed -this can be accomplished via several means, including observation, counting, workshops, interviews, and questionnaires.

The outcomes of **Process Performance** are an understanding of the key metric data, the underlying capability of the process and customers, suppliers and staff requirements for the future improved Process.

Process Review, the data and information that has been collected and analysed is reviewed and recommendations made for the improved process. Several tools, such as Cause and Effect, Pareto and Force Field Analysis can be used in this step.

The outcomes of **Process Review** include the identification of either continuous improvement activity and a process re-design project, plus the identification of any

tactical “quick wins”. The business benefits and timescales for realizing these must also be identified, together with process improvement resource allocation, performance metrics and a monitoring and reporting mechanism.

Process Change translates the prioritised process improvement mandates into an integrated programme of continuous improvement or process re-design activity. Detailed project plans with milestones, objectives, performance measures and targets, benefits, roles and deliverables must be developed, as well as a plan to manage the change and train all necessary personnel in the new process. Once the previous five steps have been implemented it is essential that the improvements that have been achieved are sustained.

In the final **Capturing the Change** step, the process improvements are integrated into the business management system, ensuring the change is reviewed, managed and built upon. Procedures should be written for the improved process, the changes, improvements and benefits communicated to all stakeholders, any training conducted, and the process and procedures regularly audited.

1.4 Effectiveness of Software Process Improvement

After the application of SPI the organization feels the competitiveness, increase in performance and innovations in the processes which show the benefits taken from the successful software processes [6] [7]. It shows the transition from one state to another. SPI focuses towards the development of the practices, improved quality of the product, reliability, productivity, and customers and employees satisfaction. The change occurs in the shape of good staff, improved technical system, organized structure and better management practices [6]. Change agents must update the skills of the current employees so that new roles and responsibilities can be understood for better career development and technical systems which contain tools and various development methodologies must also be improved in order to get better results. Organizational structure also plays an important role that how information moves with in the organization. How much interaction is done by the different groups in an organization? How activities are coordinated and managed? The organizational structure must be redesigned in such a way

that maximum communication must be made possible. The flow of information must be in all the directions and in all the departments. Three areas are most interesting in measuring the organizational effectiveness is process performance, stakeholder satisfaction and final results [7].

1.5 Benefits of Software Process Improvement

Organizations who have implemented SPI agree on the following benefits derived [8].

1.4.1 Predictability

With experience and measurements, organizations have improved their estimation of schedule and budget through time.

1.4.2 Quality

Program defects decreases as productivity increases. By adopting the SPI program in the organization the Product quality is increased consistently.

1.4.3 Customer confidence

The establishment of processes gave customers security in terms of having its projects managed professionally and not being dependent on heroes as its applications scale. Note though that your first deployment of processes can result to resistance from customers as your flexibility in accommodating concerns will now have processes to be followed. with Proper customer orientation, this can be managed accordingly.

1.4.4 Better employee morale

Employees gain higher morale in the process as they are much guided in their work through policies, training programs, and well-documented plans. Being part of organization that practices SPI gives credibility and prestige to its people as well [9].

1.6 Organization of Thesis

This section introduces the organization of thesis:

The first chapter explains the introduction to Software Process Improvement. It gives the basic idea about software processes and explain general steps for process improvement. Effectiveness and advantages of process improvement is also discussed. It gives the motivation for the thesis.

The second chapter briefly explains about the software process improvement models which are generally adopted by the software Industries. Some factors are explained which affect the software process improvement from the organization point of view. Weakness and failures reasons for software process improvement are also discussed.

The third chapter gives the idea about why the small scale industries are not adopting software process models and why they have to adopt them. Challenges and related solutions are also discussed for small scale organizations.

The fourth chapter covers the problem statement for the thesis.

The fifth chapter gives the solution to the thesis problem. SPIMS (Software Process Improvement model for small scale Industries) model is proposed.

The thesis is been concluded in the sixth chapter, and the Future work that are related to thesis topic.

Chapter 2

Software Process Improvement

2.1 Software Process Improvement Models

The explosion of technological development has led organizations to adopt new technologies at an increasing rate. The various models that are related to SPI are briefly described.

2.1.1 IDEAL Model

The IDEAL Model provides an effective approach to adopt improved software engineering processes, methods, and tools. IDEAL provides a usable, understandable approach to continuous improvement by outlining the steps necessary to establish a successful improvement program [10]. The goals of the IDEAL Model are to continuously improve the ability to implement change. The model consists of five phases.

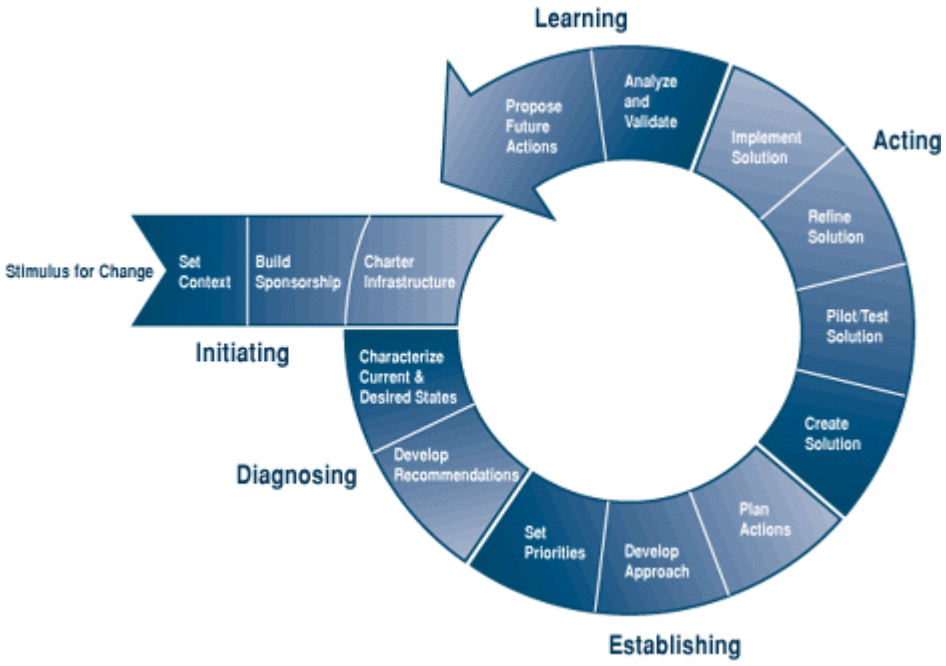

Figure 2.1: IDEAL Model [10]

The Initiating Phase

In initiating phase Critical groundwork is completed .The business reasons for undertaking the effort are clearly articulated. In this phase effort's contributions to business goals and objectives are identified, as are its relationships with the organization are other work.

The Diagnosing Phase

The diagnosing phase builds upon the initiating phase to develop a more complete understanding of the improvement work. During the diagnosing phase two characterizations the current state of the organization and the desired future state are developed.

The Establishing Phase

The purpose of the establishing phase is to develop a detailed work plan. Priorities are set that reflect the recommendations made during the diagnosing phase as well as the organization's broader operations and the constraints of its operating environment. Finally, specific actions, milestones, deliverables, and responsibilities are incorporated into an action plan.

The Acting Phase

The activities of the acting phase help an organization implement the work that has been conceptualized and planned in the previous three phases. These activities will typically consume more calendar time and more resources than all of the other phases combined

The Learning Phase

The learning phase completes the improvement cycle. IDEAL Model is to continuously improve the ability to implement change. In the learning phase, the entire IDEAL experience is reviewed to determine what was accomplished, whether the effort accomplished the intended goals, and how the organization can implement change more effectively and/or efficiently in the future [32].

2.1.2 PSP Model

The personal software process (PSP) has been developed by the Software Engineering Institute (SEI) to address the need for process improvement in small organisations and small project teams. The PSP is a self-improvement process designed to help individual engineers to control, manage, and improve the way they work. It is a structured framework of forms, guidelines, and procedures for developing software [11].
The basic principles of the PSP are as follows:

- Software professionals will better understand what they do if they define, measure will then have a defined process structure and measurable criteria for evaluating and learning from their own and others' experience.
- With this knowledge and experience, they can select those methods and practices that and track their work.
- They best suit their particular tasks and abilities[13].

2.1.2.1 The PSP Framework

The PSP has a maturity framework much like that of the CMM. The figure below shows the PSP framework. Each process improvement phase is briefly described; the description concentrates on the design, code and test phase; however, these phases are only examples. The PSP applies not only to them but also to almost any other aspect of the software process, including requirements specification, product maintenance, test planning, and documentation development.

PSP0: The Baseline Process

PSP0 The first step in the PSP is to establish a baseline that includes some basic measurements and a reporting format. This baseline provides a consistent basis for measuring progress and a defined foundation on which to improve should be the process currently used to develop software, but enhanced to provide measurement. The PSP0 is enhanced to PSP0.1 by adding a coding standard, size measurement and the process improvement proposal (PIP).

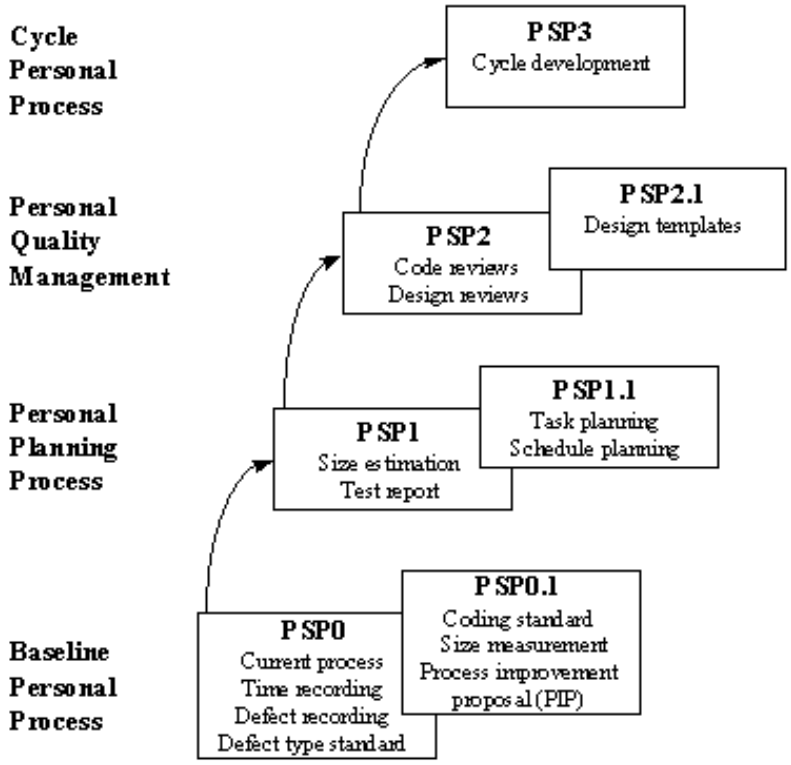

Figure 2.2: PSP Framework [12]

PSP1: The Personal Planning Process

In PSP1.1 task and schedule planning are introduced. Once an individual's performance rate is known, he or she can plan the work more accurately, make commitments more realistic, and meet those commitments more consistently.

PSP2: Personal Quality Management

PSP2 adds review techniques to PSP1 to find defects earlier when they are least expensive to fix. This is done by gathering and analyzing the defects found during compiling and testing from earlier software programs.

PSP3: Cycle Personal Process

In PSP3 an iterative approach is introduced for developing larger programs. The cyclic PSP3 process effectively scales up to larger programs only as long as each successive increment is of high quality [12].

2.1.3 SPICE Model

SPICE stands for Software Process Improvement and Capability determination. Process improvement has the objective of changing or optimizing processes for greater effectiveness to achieve gains in product quality and productivity. Capability determination however is concerned with assessing an organization or project in order to determine risks to the successful outcome of a contract, development or service delivery [14].

The objective is to assist the software industry to make significant gains in productivity and quality, while at the same time helping purchasers to get better value for money and reduce the risk associated with large software projects and purchases [15].

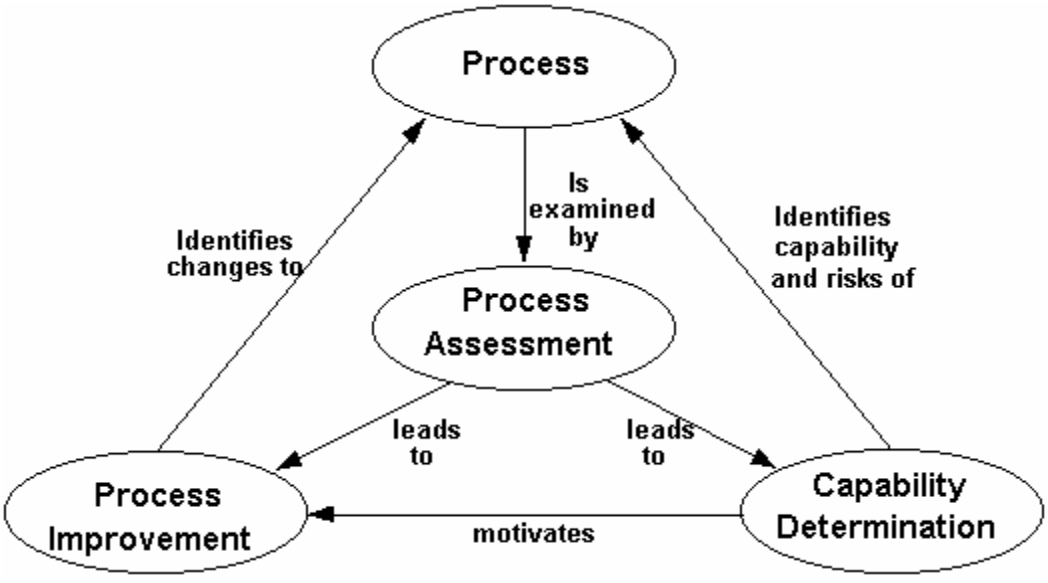

Figure 2.3: SPICE Model [15]

The practices are addressed by capability levels. The capability levels defined within SPICE are:

i. Incomplete Process:

The process is not implemented, or fails to achieve its defined process outcomes.

ii. Performed Process

The implemented process achieves is defined process outcomes.

iii. Managed Process

The previously defined performed process now delivers work products that fulfill expressed quality requirements within defined timescales and resource needs.

iv. Established Process

The previously defined managed process are performs using a defined process that is based upon good software engineering principles and is capable of achieving its defined process outcomes

v. Predictable Process

The previously defined established process now performs consistently within defined limits to achieve its defined process outcomes.

vi. Optimizing Process

The previously defined predictable process now dynamically changes and adapts to effectively meet current future business goals [16].

2.1.4. CMM Model

The Capability Maturity Model (CMM) developed at the Software Engineering Institute is based on the premises that maturity indicates capability and to obtain continuous process improvement it is much better to take small evolutionary steps rather than revolutionary innovations. it aims at guiding software organisations in selecting process improvement strategies by first determining their current process maturity before identifying their organisation's critical quality and process improvement issues [18].

These five developmental stages are referred to as maturity levels, and at each level, the organization has a distinct process capability. By moving up these levels, the organization’s capability is consistently improved.

Level 1: Initial Level

Level 2: Repeatable Level

Level 3: Defined Level

Level 4: Managed Level

Level 5: Optimizing Level

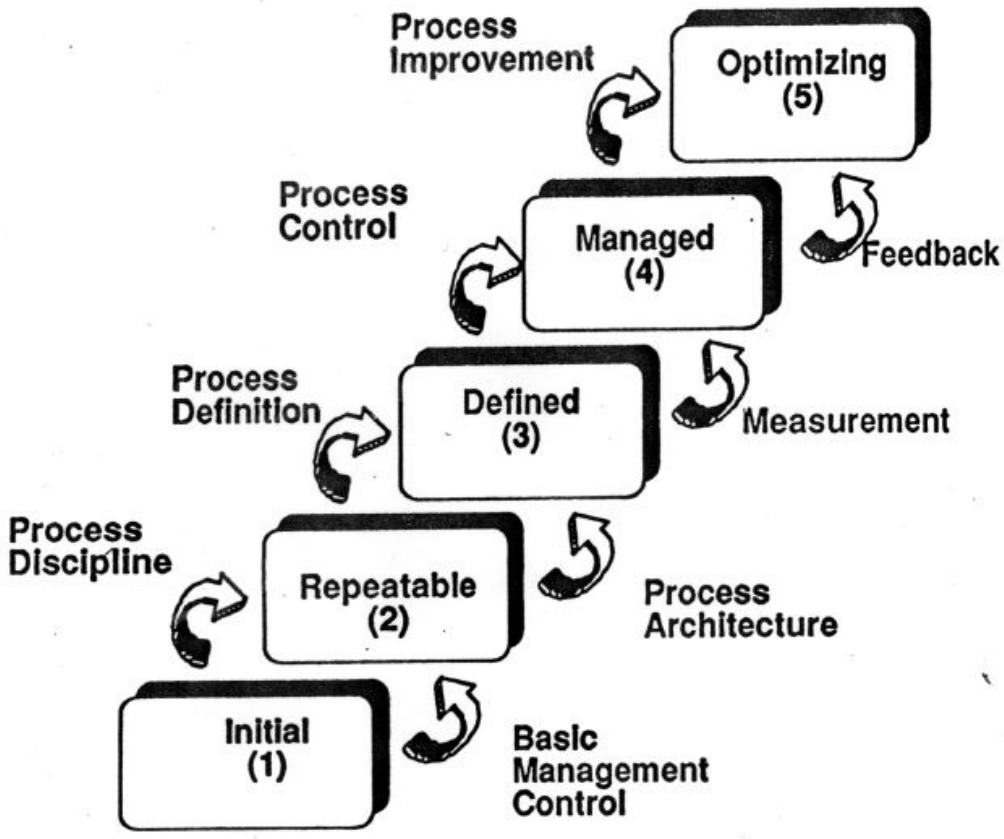

Figure 2.4: CMM Model [17]

2.1.4.1 The Five Maturity Levels

The following characterizations of the five maturity levels highlight the primary process changes made at each level:

i. Initial

The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.

ii. Repeatable

Basic project management processes are established to track cost, Schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

iii. Defined

The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

iv. Managed

Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

v. Optimizing

Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies [17].

2.1.4.2 Key Process Areas of CMM

Except for level 1, each maturity level is decomposed into several key process areas that indicate the areas an organization should focus on to improve its software process. Key process areas identify the issues that must be addressed to achieve a maturity level. Each key process area identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability. By definition there are no key process areas for level 1.

The key process areas at level 2 focus on the software project's concerns related to establishing basic project management controls, as summarized below:

i. Requirements Management (RM)

Establish a common understanding between the customer and the software project of the customer's requirements that will be addressed by the software project.

ii. Software Project Planning (PP)

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

iii. Software Project Tracking and Oversight (PT)

Establish adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans.

iv. Software Subcontract Management (SM)

Select qualified software subcontractors and manage them effectively.

v. Software Quality Assurance (QA)

Provide management with appropriate visibility into the process being used by the software project and of the products being built.

vi. Software Configuration Management (CM)

Establish and maintain the integrity of the products of the software project throughout the project's software life cycle

The key process areas at level 3 address both project and organizational issues, as the organization establishes an infrastructure that institutionalizes effective software engineering and management processes across all projects, as below:

vii. Organization Process Focus (PF)

Establish the organizational responsibility for software process activities that improve the organisation's overall software process capability.

viii. Organization Process Definition(PD)

Develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long-term benefits to the organization.

ix. Training Program(TP)

Develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently.

x. Integrated Software Management (IM)

Integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets.

xi. Software Product Engineering (PE)

Consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.

xii. Intergroup Coordination (IC)

Establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.

xiii. Peer Reviews (PR)

Remove defects from the software work products early and efficiently. An important corollary effect is to develop a better understanding of the software work products and of the defects that can be prevented.

The key process areas at level 4 focus on establishing a quantitative understanding of both the software process and the software work products being built, as below:

xiv. Quantitative Process Management (QP)

Control the process performance of the software project quantitatively.

xv. Software Quality Management (QM)

Develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals.

The key process areas at level 5 cover the issues that both the organization and the projects must address to implement continuous and measurable software process Improvement as below:

xvi. Defect Prevention (DP)

Identify the causes of defects and prevent them from recurring.

xvii. Technology Change Management (TM)

Identify beneficial new technologies (i.e., tools, methods, and processes) and transfer them into the organization in an orderly manner.

xviii. Process Change Management (PC)

Continually improve the software processes used in the organisation with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development [19].

2.1.5. BOOTSTRAP Methodology

BOOTSTRAP methodology can be applied to small and medium size software companies or software departments within a large organization. A new release (Release 3.0) of the BOOTSTRAP methodology has been developed to assure conformance with the emerging ISO standard for software process assessment and improvement [20].

The BOOTSTRAP methodology has the following objectives:-

- To provide support for the evaluation of process capability against a set of recognized software engineering best practices.
- To include internationally recognized software engineering standards as sources for identification of best practices.
- To identify, in the assessed organization, process strengths and weaknesses.
- To support improvement planning with suitable and reliable results.
- To support the achievement of the organization's goals by planning improvement actions [22].

2.1.5.1 The Reference Model

The reference model describes the typical software development organization. The reference model is also called the Process model. It is used as a reference and a guide to identify good software engineering and management practices. It is based on ISO 12207 and ISO 15504. The processes of the typical software organization are categorised into three categories: Organization, Methodology and Technology (process categories). The process model is shown in figure [21].

The BOOTSTRAP process model defines processes and capability levels. Process capability is measured based on the following capability levels:

Level 0: Incomplete Process

Level 1: Performed Process

Level 2: Managed Process

Level 3: Established Process

Level 4: Predictable Process

Level 5: Optimizing Process

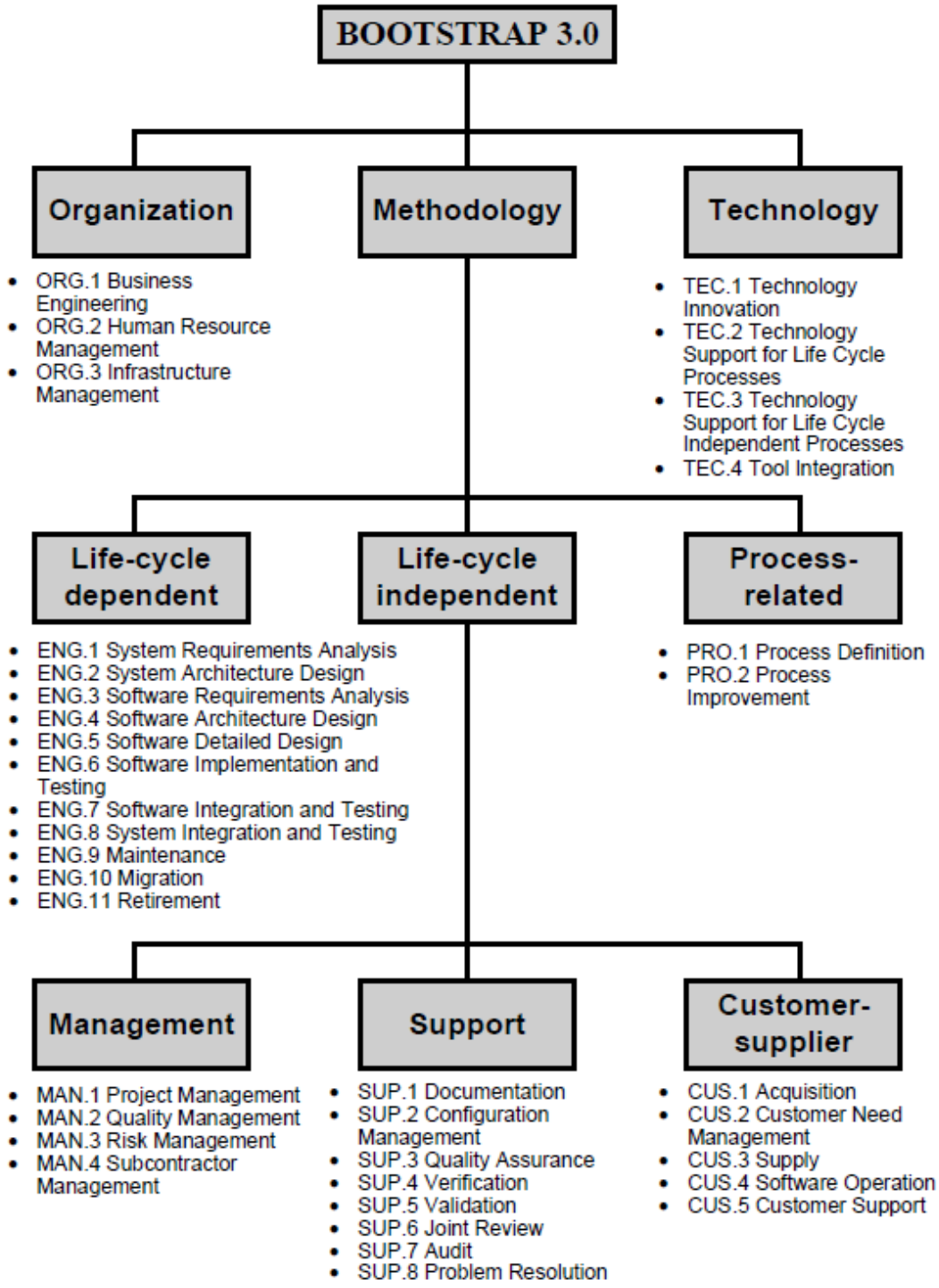

Figure 2.5: BOOTSRTAP MODEL [22]

2.1.6 SPI Implementation Model

The structure of SPI implementation model is built upon the following elements

- SPI Implementation Phase Dimension
- SPI Implementation CSFs Dimension

2.1.6.1 SPI Implementation Phase Dimension

Using the content analysis of the recorded interviews, we have identified six phases for the implementation of SPI programmes. we briefly describe each phase in turn and in the appropriate sequence. Figure shows SPI implementation model [23]

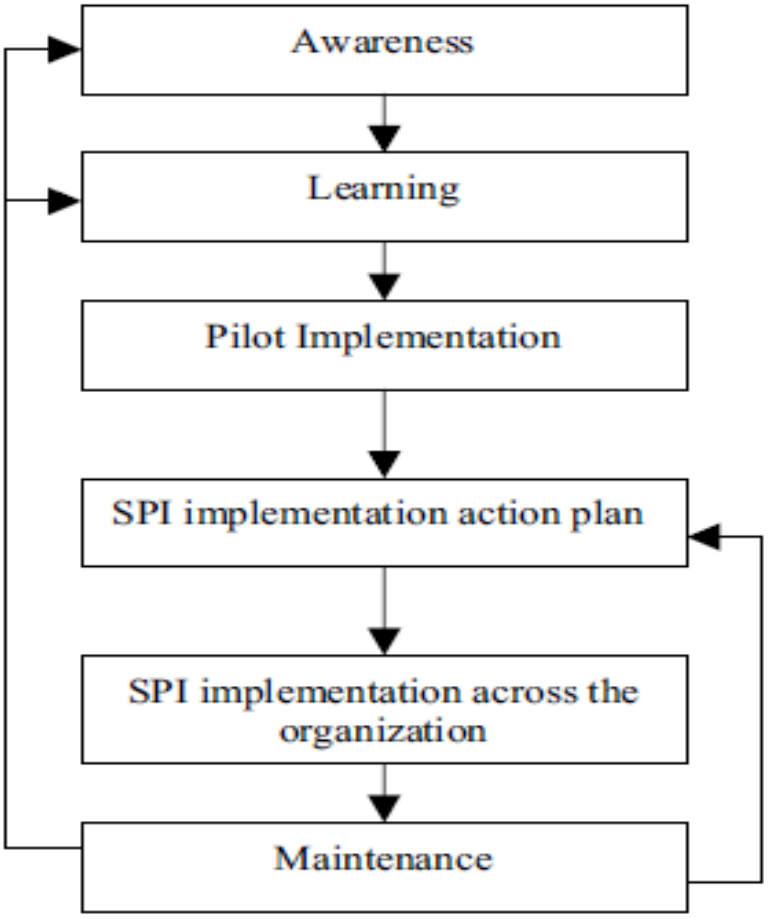

*Figure 2.*6: SPI Implementation Model [23]

i. Awareness

Practitioners felt the need for awareness of SPI programmes in order to fully understand the benefits of SPI. Practitioners said that as SPI implementation is the process of adoption of new practices in the organization, it is very important to conduct high-level sessions for practitioners in order to provide them sufficient knowledge of SPI. Practitioners suggested involving all the staff members in these awareness programmes.

ii. Learning

Learning appears as an important factor for SPI implementation success. For learning, practitioners emphasized training in SPI skills in order to achieve mastery of its use. This phase involves equipping the practitioners with the knowledge of the critical technologies which are required for SPI. SPI can only be successfully implemented if staff members have enough understanding, guidance and knowledge of all the SPI activities [24].

iii. Pilot Implementation

Practitioners advised to first implement SPI programs at a low level and see how successful it is within a particular department. This is also important for practitioners in order to judge their SPI skills in the pilot implementation. This is the phase where practitioners can decide how much resources, training and commitment is required in order to implement SPI practices across the organization.

iv. SPI Implementation Action Plan

Practitioners stressed the need for proper planning and management. They said after pilot implementation a proper plan with activities, schedule, allocated resources, responsibilities, budget and milestone should be designed. This plan should be based on the results and experiences of the pilot implementation. SPI implementation without planning and project management leads to chaotic practices [26].

v. Implementation across the Organization

After proper planning and using the pilot implementation experience, practitioners suggested to implement SPI practices in other areas/departments of the organization in order to have uniform development approach and maturity across the organization. It is also important to illustrate the results of pilot implementation to different departments in order to get support and confidence In order to avoid risks and to implement SPI programmes more effectively, practitioners suggested project-by-project implementation. This is because each project experience can be viewed to determine what was accomplished and how the organization can implement SPI programmes more effectively for future projects. Practitioners emphasised that senior management commitment plays a very important role in this phase. They also suggested providing sufficient resources for SPI implementation in this phase.

vi. Maintenance

The important theme in maintenance is to continuously monitor and support the previously implemented SPI. activities. Practitioners suggested continuing awareness and training programmes to be incorporated into maintenance activities as practitioners often switch jobs. SPI efforts do not have long lasting effects because practitioners often slide back to their old habits . It is therefore very important to continuously provide them with feedback, guidance, motivation and reinforcement to stay involved in the improvement effort [25].

2.1.6.2 SPI Implementation CSFs Dimension

Different studies have confirmed the value of the CSF approach. A review of the CSF literature reveals that the concept has not been employed to any great degree in research on the topic of SPI implementation. Implementation of SPI programmes require real life experiences where one learns from mistakes and continuously improves the implementation process. CSFs are often identified after the successful completion of certain activities. Hence these factors are near-to real life experiences. Therefore, we believe that CSFs approach can also be useful in the implementation of SPI .

We have divided CSFs and critical barriers among different phases of implementation model as shown in Table 1. The basis of this division is the perceived coherence between the CSFs and critical barriers identified. It should also be pointed out that these factors and barriers are not necessarily mutually exclusive and there may be a certain degree of overlap among them.

Table 1 suggests that practitioners should consider these CSFs and critical barriers in order to successfully implement each phase of the SPI implementation model because we have more confidence that a factor does indeed have an impact on SPI implementation if it is critical in the literature [27].

Phase	CSFs	Critical Barriers
Awareness	Senior management commitment, Training and mentoring, Staff involvement	Organizational politics
Learning	Senior management commitment, Training and mentoring	Time pressure
Pilot Implementation	Senior management commitment, Creating process action teams, Experienced staff	Inexperienced staff
SPI implementation action plan	Senior management commitment, Clear and relevant SPI goals, Assignment of responsibility of SPI, Experienced staff	SPI gets in the way of real work, Inexperienced staff
SPI implementation across the organization	Senior management commitment, Staff time and resources, Staff involvement Experienced staff	Organizational politics, Time pressure, SPI gets in the way of real work, Inexperienced staff
Maintenance	Senior management commitment,Reviews , Training and mentoring	Staff turnover

Table 1: SPI Implementation CSFs [27]

2.2 Factors which affect the Software Process Improvement

Be aware of your organization's current culture

One of the significant forces that affect the success of your process improvement efforts is the culture of your organization. organizations with cultures that are positive toward process improvement are likely to want to supply a quality product with reasonable business returns, have middle managers that are willing to set and work toward targets of meeting your organization's needs and business goals.

Expect to change your organization's structure and culture

Process, organizational structure, and corporate culture all go hand-in-hand – change one and you will affect the others.

Keep it simple

A common mistake that organizations make is to over specify the processes that they intend to follow. Never forget that your goal is to produce working software that meets the needs of your user community and that your staff likely has a pretty good idea of how to do this although they could help with a little guidance from time to time. Give them just enough guidance for their needs. An example of a well defined, yet simply defined, process is the Agile Unified Process (AUP).

Align your software process with business goals and objectives

You could have the best process in the world, but if it doesn't meet your **organization's** goals then it doesn't matter. Do you intend to build a portfolio of applications that integrate with, and build upon, one another? If so then portfolio management is important. Do you intend to sell shrink-wrapped software to be used by millions of users? If so then architecture may be less important to you in favor of getting a product to market quickly.

Regularly hold retrospectives

A retrospective is a process improvement meeting where you ask four fundamental questions: What did we do well that if we don't discuss we may forget? What did we

learn? What should we do differently next time? What still puzzles us? Retrospectives can be simple 15 minute discussions or formal meetings over a day or two. The goal is to learn from your experiences.

.

There's a serious difference between "lessons learned" and "lessons indicated"

The result of a **retrospective** is a collection of lessons indicated, they don't become "learned" until you actually improve your process.

Keep the real goal in mind

Organizations that keep the end goal in mind – that of developing, maintaining, and supporting software that fulfills the needs of their user community – will be successful with implementing software processes. Those that follow processes simply for the sake of doing so are likely to fail.

Treat process improvement like a project

Have an experienced project manager, ideally someone with experience in both process-oriented and object-oriented development. Define the requirements for your processes, model them, implement them, test them with, and then improve the processes.

Communicate your plan

It is essential that you let others know what are you doing, why are you doing it, the business case for it, success stories, etc. Keeping them informed on how things are going, even if you are encountering difficulties, will help get them (and keep them) on board. There are various options for this: a newsletter, a web site, periodic emails to key personnel etc.

2.3 Failures of SPI

Management's commitment [27] and understanding of the tasks play an important role in the success or failure of process improvement activities which may also occur due to the:

- Misallocation of the resources and the man power or majorly due to the multiple shared or insufficient resources. It also reflects the poor resource planning.
- Unskilled or less skilled staff can also affect the business orientation. Orientation must in the hands of experts.
- Mismanagement in the practices and in staff performances. The management wants to get rid of assessment as early as possible.
- Lack of software management skills. Incapable technical staff leads towards this state.
- Lack of business orientation of the actors and of the programs keeping in view the business strategies and goals.

A company decided to initiate software process improvement procedures. Higher complexity means higher risk in the business because of the new upcoming challenges and low quality product. The following principles were made the basis for the initiative of process improvement [27].

i. CMM will be used as the basis of assessing the business and software perspectives.

ii. Self assessment and training of the staff for the better SPI initiative.

iii. Steering committee must play an active role in the beginning.

iv. Proper budget must be allocated to initiate and run the improvement activities.

v. SPI actions must be shared by the executives.

vi. Performance must be evaluated to see the impact of change inside organization.

vii.The awareness from the importance of SPI must be highlighted so that the involvement of the executives can be introduced. It not only helps the processes but also to the quality factors. Research and development department must also participate in order to fulfil the commitment. The improvement factor must be shown in order to get the confidence of the organization and of the management. Historical data must be given high importance to compare the quality, schedule and cost of the product so that the improvement of the processes can be judged. Moreover, defect detection and defect prevention must always be given equal importance.

viii. The large projects can give better revenue if they are properly managed with improved procedures. The risk factor must be identified in order to avoid the ambiguities.

Focused efforts are required so that specialized efforts can generate better results. New set of well defined practices must also be introduced to show the improvement in the current processes so that new projects can show better results. External consultants can be hired to take the assistance to perform the software process improvement if internal expertise of the staff is low or poor.

ix. The management must be given a good introduction of the SPI. It must be shown all the pros and cons and the long run benefits to the organization. The complete process of improvement must be explained to management from execution to results so that the employee satisfaction can be achieved. External experts can also help and can better recommend the changes in the processes. The project leader must be given proper training of the basic principles of the techniques for the better execution of the practices. The people who have already experienced the implementation of the process improvement can also be included to get more benefits from process improvement techniques. Software and business process improvement might be one and the best way to compete better in the highly competitive markets in the future [26] [27].

Weaknesses and failures [26] found in SPI.

i. Weaknesses concerned with cost

- Instability of requirements is an important factor.
- Knowledge of cost deposition and distribution.

ii. Weaknesses concerned with Requirement Management.

- Changed request if it is not handled properly.
- Impact analysis is not done properly by the staff or by the responsible team.
- Change decisions at wrong time by the team or team members.
- Absence of suitable processes and proper process measurements.

iii. Weaknesses concerned with Project Management.

- Improper project monitoring.
- Unavailability of historical data for analysis.
- Lack of information updating and sharing.

- Improper risk analysis by the in competitive employees.
- Corrective actions are not taken in time as required.
- Poor planning i.e. resources, budget, time, cost etc

Chapter 3

SPI: Challenges and Solutions

3.1 Reasons for small scale organization apprehension to adopt Software Process Model

Small software organizations may be apprehensive to adopt SPM because of fear of the models being too costly to implement in money and manpower. It is also a general concern that the models simply aren't meant to be implemented in an organization of their size – that the models simply don't scale down. Below are common reasons why small organizations often choose to adopt non SPM methods [31].

3.1.1 Money

Small organizations may first balk at the seemingly prohibitive cost of adopting SPM. Researchers suggests that the cost to implement the Capability Maturity Model is between $490 and $2,004 per person with the median being $1,475 and that achieving the next level of the Capability Maturity Model can cost upward of $1,000,000. In addition, software product assessment can take between $45,000 and $100,000. Meeting the goals of some key processes can be financially taxing and it may be necessary to tailor the SPM's key process areas [28] [29].

3.1.2 Manpower

The perception is that SPM requires vast amounts of people who each have their own specific role in carrying out individual model component requirements. Small organizations see the adoption of SPM as a luxury that requires many employees in order to effectively achieve the desired results and may be more inclined to hire a single professional experience in SPM rather than train the employees from the ground up due to cost associated with training the employees and with production cycle time loss resultant in training. Training staff versus hiring new employees may seem more cost prohibitive and in either case it may not seem economically feasible [30].

3.1.3 Model Scale

SPM often (or seemingly) do not scale from the large organizations for which they were intended to small organizations who may also benefit from them. The models may require separate entities such as configuration management and software quality assurance both of which are expensive to organizations with limited resources. Small organizations simply do not have the assets necessary to fulfill such a large scale requirement [30].

3.1.4 Time

With limited staff, small organizations feel that their time will be better served working through the projects and trying to learn from their mistakes rather than wasting valuable time with employees in training. Extraordinary amounts of time will likely need to be devoted to SPM adoption. The initial time for training and adoption of the IDEAL model can be 2,100 person hours and $50,000. Software process improvement is a slow course of action and just transitioning from the first level of the Capability Maturity Model to the second is to take between 21 and 37 months (with slight time improvement in transition between higher levels of the model) [30] [32].

3.1.5 Customer Rules

Many small organizations adopt the concept of “customer rules”. They believe that adhering to a set standard lends them to being to rigid or inflexible to changing customer demand and in turn the organization may adopt the customer's own process. Some organizations even fear that SPM adoption injects too much bureaucracy into the system. Customer satisfaction is every businesses goal and small organizations often feel that they can provide that satisfaction by providing a product modeled around the customer's own rules rather than using what they perceive to be a rigid, impersonal, and red tape laden system [31].

3.2 Why Small organizations should adopt Software Process Models

Despite there being the fundamental size differences between large and small organisations, it is possible for small organizations to overcome the hurdles that SPM presents. It has been found that with proper application and adoption of a SPM, small organisations can achieve results similar to larger organisations as well as even surpassing the larger organisation's results. Small organisations can implement software process improvement at least as well as large organisations. In fact, small organizations process certain qualities which make them even more readily able to adopt the SPM [31].

3.2.1 Inherent Adaptability of Small Organizations

Ease of SPMM adoption is made possible by the inherent flexibility of small organisations. A small organization may be better suited to adopt SPM because of the relative lack of red tape and politics which is prevalent in much larger organisations. Because of the decreased obstacles and the tendency for smaller organisation's rapid change adaptability, SPMM adoption is a generally easy leap to make as compared to larger organizations [30] [32].

3.2.2 Ability to Manipulate the Models

Software startups must rapidly enter the market and while immediately adopting a mature SPM may not be possible it's vital to adhere to some standards as soon as possible. For small organisations, formal methods should be mixed with informal practices for best results. Maturity requires repeatability and resources – some characteristics startups typically don't have. Being a small organization, it may be necessary to manipulate the SPM to fit the environment and when possible it is prudent to fully adopt a mature process model. Although at first it is beneficial to simply begin with adopting a process, remaining flexible throughout the process adoption, and properly defining the process should be the ultimate goal. In addition, generalizing the development aspects, tools, protocols, etc. that may be used and determining the reuse from those experiences will help towards gaining full SPM adoption [31] [32].

3.2.3 Formalize Goals

Adherence to SPM is a means of structuring a software project and clearly defining desired goals. Formalized software design and development goals allow thoroughness of action and follow through by structuring the intentions. They are created in a manner by which the software developers can generate the necessary components efficiently as well as lessening the burden inherent in guessing. The organization is also able to keep valuable resources from having to determine the scope and intent of the project after the development phase has begun [29].

3.2.4 More Readily Predict Issues

Greatly enhanced levels of predictability can be achieved by SPM adoption. One of the most significant benefits of adopting the SPM is being able to gather data about the past and current projects and using that data to apply a set of metrics which allows the organization to measure weaknesses, ultimately being more readily able to predict where failures may occur. Predictability alone is an incredible asset from which a small organization can tremendously benefit

.

3.2.5 Increase Workflow

The use of SPM increases productivity and workflow by providing a standard by which the organization can abide. SPM are the assembly line of the software industry – they provide the team of developers a structure by which they may efficiently and effectively deliver products. A structured environment is extremely beneficial is helping to prevent unnecessary deviation found in organisations who adhere to no SPM.

3.2.6 Achieve Better Marketability and Competitiveness

Successful implementation of SPM includes relationships with larger companies with greater process maturity and also becoming a member of an organization that helps with software process improvement. Small organisations that adopt SPM are more marketable and, from that, more competitive. Growing organisations require the operational efficiency that SPM provides and that efficiency is seen by the market as maturity and

capability. SPM adherence is expected by other or larger entities. Organisation who place bids on contracts are given preference if they show adherence to SPM. Adopting and adhering to a model is a way of telling potential customers that the resulting product is one worthy of emerging from the crowded marketplace.

3.2.7 Improve Customer Satisfaction

Of course, the ultimate goal of an organization is to remain in business, whether it is for money or altruistic purposes. The way to achieve that is through customer satisfaction and small organizations can significantly increase the level of customer satisfaction by adopting SPM. SPM adherence shows the customer that the organization in serious in its venture and assures the customer that the product meets quality assurance standards. Altogether it can lead to improved customer satisfaction before they even have the final product.

3.3 Challenges in Small Organizations as compared to Large Organizations

Before talking about the software engineering challenges in small companies, I would like to emphasize that the scale of the software companies is "small". Larry argues that it is the business goals, not the organization's size that matter in software development. Though given different characteristics of small and large companies, if they have the same business goal, then, they can apply the same software engineering techniques. Philosophically, that might be true under some particular conditions, because both of the small and large firms want to make benefits in marketing and to maintain good relationships with their customers. However, practically, the ways to achieve similar business goals can be greatly varied from large companies to small ones. This makes the whole story be different to small companies [33].The size of organization has influences on the choice of software development strategies and the use of SE technologies, because leaders have to take the constraints of staffing and budgets into consideration. According to this situation, three main software engineering challenges faced by small companies are described as follows:

Firstly, using toolkits, the copyrights of which are reserved by other commercial entities, can be costly. To organize software process, project leaders need the help from lots of development and management tools. In large firms, they may have professional teams to develop and maintain their own development platforms and secondary toolkits, which can improve the whole work efficiency. On the contrary, this is not affordable to small software companies. Though small companies need tools to communicate with customers and to cover every step in software process, the high cost in purchasing new secondary toolkits will make the project easily over its original estimation [32].

Secondly, complex but helpful measurement can be time-consuming for small software companies. The benefits of measurement are usually found in analysing the effects on software process by improvement efforts. The large firms, of course, with enough resources, have published their practical experiences in apply metrics programmes in software process improvement. Even though, the managers and developers in small software companies are reluctant to measurement, partly because they doubt whether the metrics programmes can lead them to success after spending extra time on measurement.

Thirdly, small software companies suffer from the lack of real-world publications from similar companies describing efforts on an improvement initiative. Adopting internationally accepted software process practices is essential for software companies to compete in the global software development market. Although more and more studies on successful initiatives are available in recent years, they still seem to be restrictive.

3.4 Solutions to Challenges

3.4.1 Open Source Tools used in Software Process

As the large software organisations, small organizations aim to gain all the benefits through exploiting the software engineering practices. However, unlike large companies, small companies face with the lack of staff to develop functional specialties to perform complicate tasks supporting there products implementation. The software process designed needs to be lightweight, low cost in training and suitable for rapid development. Therefore, most of the tools used in the process should be open source. In Ken and Bill's

open source approach applied to software development, particularly for long-term project, the software process is conducted into five parts, namely, communication and documentation, revision control, building management, testing, and release process. Fortunately, more and more open source tools covering almost all parts of software process are available these days. During software development and maintenance, effective communication and sufficient documentation can lay a solid foundation in software process. The role of communication is also stressed with its influences on the perception of the innovation. In recent years, Wiki has become one of most popular communication tools in the world. The most famous application of Wiki technology is the Wikipedia. To introduce Wiki into software process, especially in communication aspect, has become a first place option for many project managers. Wiki, which is software that is used more like a platform to allow users to create, edit, link, and organize the communication contents collaboratively, is a preferable place to put Q&As issued by either customers or developers, and to record new sparkling ideas freely .It helps to narrow the communication gap between developers and customers, even between developers themselves. As said, the contents of Q&As and important discussions can be taken into documentation if needed. Therefore, after a long time, less important points will be missed or forgotten than ever before. The improved documentation helps developers to understand and manage their work, and meet the customers' needs for accurate information [33].

Revision control is essential not only for large software companies but for small ones as well. No matter how small the project will be, the project teams need the Revision Control System (RCS) to store, retrieve, log, identify and merge revisions automatically. With the RCSs, developers can easily track changes in codes and documents at any time and manage file versions. Concurrent Versions System (CVS) and Subversion are two open sources revision control tools widely used now a days.

The Trac system can be a good example of the open source tools used in software process. Trac combines information between wiki content, revision control, and a computer bug database by allowing wiki mark up in issue descriptions and commit

messages, creating links and seamless references between bugs, tasks, change sets, files and wiki pages. Besides, it serves as a web interface to a revision control system, like Subversion [32].

3.4.2 Simplified Measurement

The goal of measurement is to improve software process, and almost all software companies can benefit from the improved software process. Thus, theoretically, software companies no matter whether they are small or large should warmly welcome measurement and metrics programmes. Actually, the story is rather opposite to that ideal thought. The software developers, especially from small companies, tend to greet measurement activities with scepticism before they see true profits from metrics programmes. After the researchers had an insight into the doubts of software developers, they found the reasons as follows: the measurability of software work, the usefulness of data collection, the fears of being controlled by their employers, and the extra workload spent on measurement, which is the most considerable thing. For these reasons, the comprehensive, complicated metrics programs applied in large firms are meaningless for those small software companies [33] [35].

In order to eliminate the doubts of software developers, new, simple, quantitative, small-scale metrics programs are required. Moreover, the metrics programmes need to be individualized for different companies with high diversities in their characteristics and key objectives. Examples for varied key objectives of small software companies are to reduce dependency on the chief developer by having all team members' work on all parts of their product, to maximize the accepted change requests handled in a certain period, to minimize the time spent on handling customer change requests, etc .The next step is to gather data from all possible resources. Since the developers may have residual doubts about measurement benefits and increase in workload, it is better to aggregate figures from existing available data. The following step is to generate the results of measurement, which is an important step to make the whole evaluation be visible and fruitful to developers and customers. The key to successful measurement is to understand the objectives of small companies. It has been proved that technology transfer is a

process happening in social environment rather than a context-free technical matter. Given different situations and characteristics, we should adjust metrics programmes correspondently. Though small in size, the software companies with fewer than 20 developers need some basic formal routines as well. only if metrics programmes are carefully selected and the purposes of which are well dedicated, we can achieve the balance between mechanisms and documented procedures [34] [35].

Chapter 4

Problem Statement

There are number of Software Process Improvement models are available which are not readily adopted in small scale organizations. Small Scale Organizations do not find the software process models easy to implement.

Today, the software industry is one of the most rapidly growing sectors and small software development companies play an important role in economy. Many such organizations have been interested in Software Process Improvement (SPI). It has been observed that the successful implementation of SPI methodologies is generally not possible within the context of small software enterprises because they are not capable of bearing the cost of implementing these software process improvement programs. Further the proper implementation of software engineering techniques is difficult task for small enterprises as they often operate on limited resources and with strict time constraints.

Large organizations greatly benefit financially and organizationally from the adoption and adherence of software process models. The models are useful in laying out a groundwork for the software development process, for determining allocation of resources, determining a cost and time analysis, providing the customer with overall satisfaction, and ultimately generating good rapport. Large organizations prosper upon Software process model adoption and their success ought to be a model of inspiration for small organizations. Large software organizations do have the benefit of being able to adopt nearly any one of the many well known SPM because the models are typically written with large organizations in mind. However, its often the case that the smaller business choose not to adopt software process model because of the general idea that SPM adoption costs much money, manpower, time and feeling that the models do not scale down. So, the main objective of the thesis is to propose a Software Process Improvement model which is easily incorporated and adopted by small scale organizations.

Quality and Process Models

Quality models and standards have been developed to improve product quality, satisfy business goals and standardize software engineering practices and procedures. Their certifications have been sought due to management improvement strategies, government policies, market competition and customer requirements. However, the implementation of quality models and standards involves time and cost overheads that are large, especially for small organizations.

The evolution of international quality model was intended to improve quality and to provide more guidelines for reducing overhead involved. ISO 9001 required establishing, documenting and maintaining a software process. The Capability Maturity Model (CMM) was intended to improve software processes using a set of recommended practices in a number of key process areas. ISO 9000-3 preceded guidance by introducing guidelines for the application of ISO 9001 to the development, supply and maintenance of software. This model do not provide much needed guidance and failed to emphasize benefits of the smallness.Tailored CMM model to suite small organizations by addressing documentation overload, layered management, limited resources and other factors. BOOTSTRAP methodology used in small and medium size organizations. BOOTSTRAP is a software process assessment and improvement that is based on benchmarking and determination of current achievement levels to assess suitable improvement program and action planning. SPICE model used for large, medium and small organizaions. The main objectives of the SPICE projects to put the first draft for software process assessment and to conduct early trials. BOOTSTRAP and SPICE models are not used by small scale industries frequently due to their large number of generalized processes.

Next chapter explains SPIMS, the proposed software process improvement framework.

Chapter 5

SPIMS Framework

5.1 Step – wise Process Improvement

The basis on which SPIMS framework is built upon is the step by step improvement which is shown in Figure 5.1.

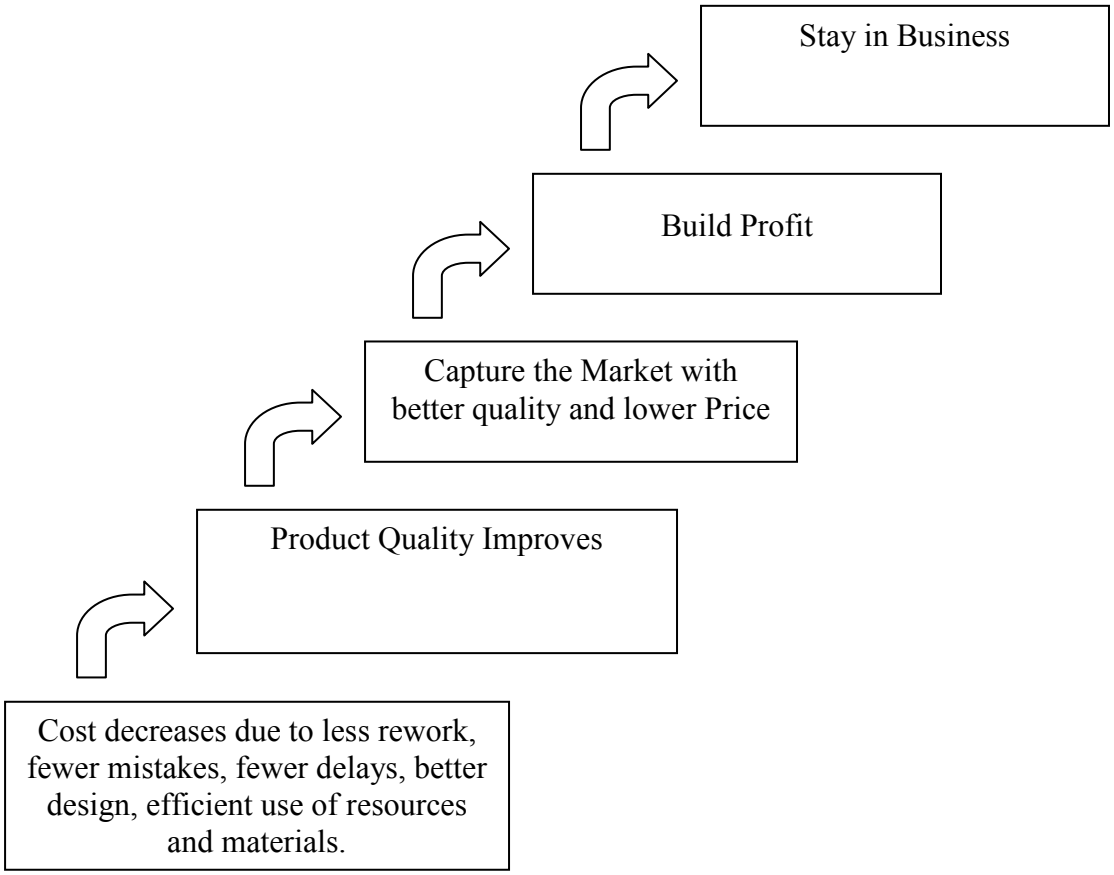

Figure 5.1: SPIMS Ladder

The first two blocks are of major concern in which the SPIMS framework is implemented which shows major cost cut offs, less rework, fewer mistakes, fewer delays, better quality products. In the SPIMS Ladder after crossing the first step, the product quality improves. Only after improving quality small scale industries can think of the market with their

better quality products. Building profit is a further step after small scale industries withstand in competition. Profit means they can put more budget and resources to improve the processes effectively. After building profit, small scale industries can go for continuous process improvement with which they can stay in business continuously. SPIMS framework provides a strong foundation for small scale industries to withstand the market scenarios. SPIMS is an iterative, specialized cum generalized model which is more or less driven by the small scale industry's own business rules.

5.2 SPIMS

SPIMS is the acronym for Software Process Improvement Model for Small Scale Industries. SPIMS is a phased as well as an iterative model which is successful in establishing specialized cum generalized set of processes at the discretion or will of small scale industries.

5.2.1 SPIMS Phases

The model has four differnet Phases with a **supporting set** of different activities or processes.

Phase 1: Customer Interaction Phase (CI)

Phase 2: Project Management Phase (PM)

Phase 3: Engineering (Implementation) Phase (ENG)

Phase 4: Business Process Optimizing Phase (BPO)

The **Customer Interaction Phase** in which developer directly interact with client for his requirement for the software Product. This is the requirement elicitation phase in which customer give his requirements as well developer also consider his implied requirements

The **Project Management** Phase which establish the project, and co-ordinate and manage its resources to produce a product or provide a service which satisfies the customer.

The **Engineering** Phase consists of processes that directly specify, implement, or maintain a system and software product.

The **Business Process Optimizing** Phase concentrates on Improvement in Buisness Processes like cost benefit analysis, quantitaive feed back, improving standard software processes, implementing improved processes effectively and also the human aspects.

The **Supporting Set** consists of activities which enable and support the performance of the other processes on a project. The supporting activities can be employed at various life-cycles and can be implemented with different Phases of Process models.

SPIMS framework in figure 5.2 is shown on next page.

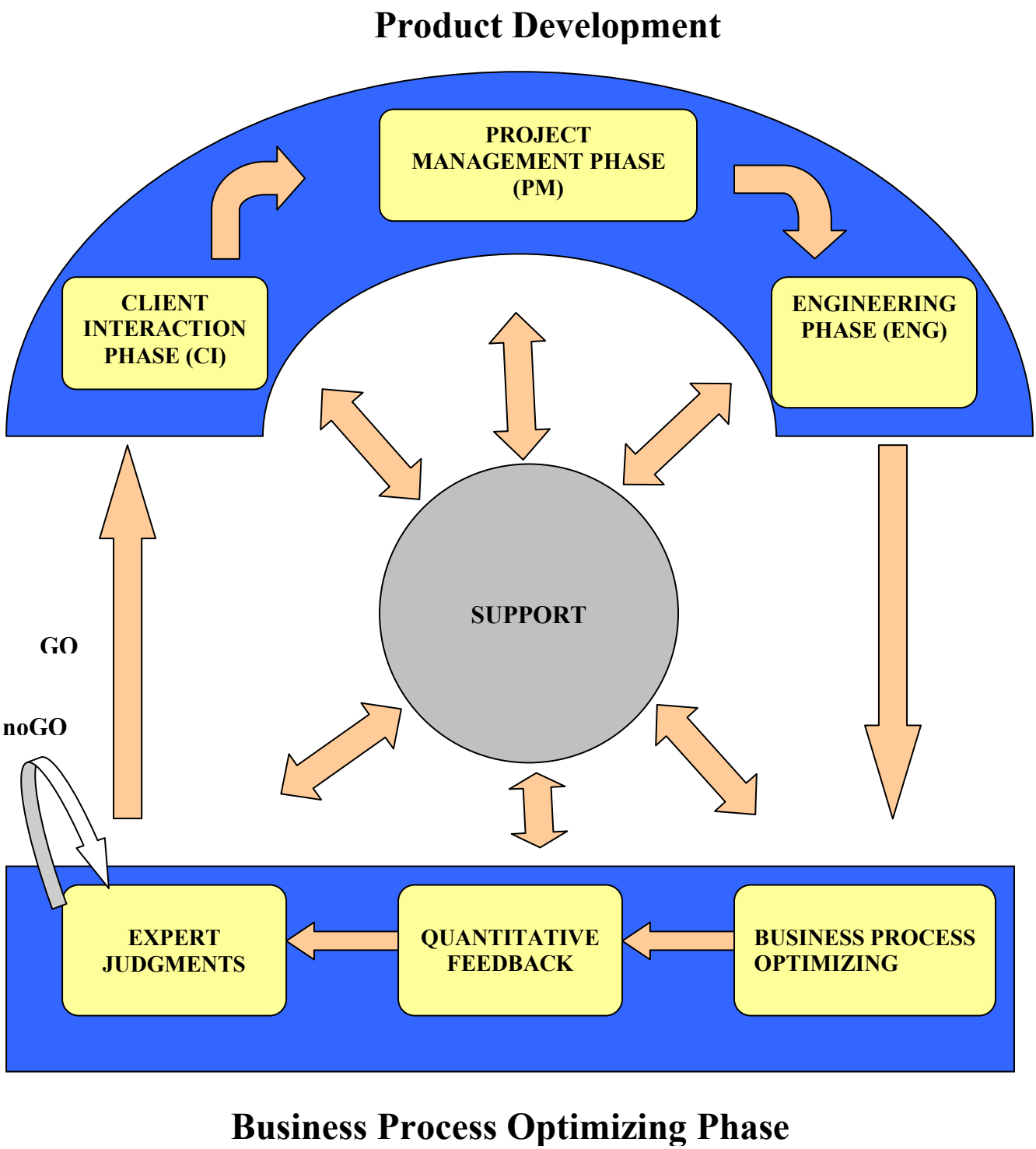

Figure 5.2: SPIMS Framework

5.3 SPIMS Phases and related Processes

Phase	Process Title
Phase 1: Client-Interaction Phase(CI)	CI 1.1 Identify Customer Needs CI 1.2 Define Requirements CI 1.3 Pricing to Win Strategy
Phase 2: Project Management Phase(PM)	PM 2.1 Establish Project Plan PM 2.2 Establish Team and Training PM 2.3 Manage Requirements PM 2.4 Project Scope Template
Phase 3: Engineering Phase(ENG) (Implementation)	ENG 3.1 Develop Software requirements ENG 3.2 Develop Software Design ENG 3.3 Implement Software Design ENG 3.4 Integrate and Validate Software ENG 3.5 Maintain System and Software
Phase 4: Business Process Optimizing Phase (BPO)	BPO 4.1 Improve Software Processes BPO 4.2 Quantitative Feedback BPO 4.3 Improving Human Aspects BPO 4.4 Put in the actions the improved processes BPO 4.5 Cost Benefit Analysis BPO 4.6 Preventing Defects
Supporting Set (SS)	Develop Documentation Perform Configuration Management Perform Problem Resolution Perform Peer Reviews Reuse(optional)

Table 3: SPIMS Phases and related processes

5.3.1 Phase 1: Customer Interaction Phase (CI)

The Customer interaction phase in which developer directly interacts with client for his requirement for the software Product. This is the requirement elicitation phase in which customer give his requirements as well developer also consider his implied requirements.

The processes belonging to the Client Interaction Phase are:

CI.1 Identify Customer Needs

CI.2 Define Requirements

CI.3 Pricing to Win Strategy

CI.1 Identify Customer Needs

The purpose of the Identify customer needs process is to manage the gathering, processing, and tracking of customer requirements and requests toward a better understanding of what will satisfy the customer.

Compare with process, "Manage requirements" PRO.4, which addresses coming to an understanding within the software project of the requirements to build to (and which become part of the development baseline). Instead, the current process emphasizes interaction with the customer to promote an understanding of customer requirements, and tracks all requirements.

CI.1.1 Eliciting Customer Requirements

This task is performed by communicating with customers and as well as with users and by making use cases scenarios. Requirement should be gathered by interviewing the customer or by brainstorming session. It should be noted that software requirements being clear, concise, consistent, understandable and non-volatile in nature.

CI.1.2 Understand Customer Expectations

Review with customers and users their requirements and requests to better understand their needs and expectations.

CI.1.3 Keep Customers Informed

Keep customers informed about the status and disposition of their requirements and requests. This may include joint meetings with the customer or formal communication to review the status for their requests including requirements.

CI 2 Define Requirements

The Purpose of define requirement is to define those requirements which are very important from customer point of view & developer point of view also. Developer should also give implied requirement for the project.

CI 3 Pricing to Win Strategy

Pricinple: Give weight to the customer requirement that is very important and have lower customer satisfaction.

This also reduces the unimportant goals and reduces cost. Pricing to win strategy makes the requirements more focused which reduces requirement analysis time.Pricing to Win solution strategy helps to define the right balance of capability as well as price and delivery of the value the customer wants.

This strategy provide comprehensive knowledge of customer's needs and Budget. The software cost is estimated to be whatever, the customer has available to spend on the project. The estimated effort depend upon the customer budget.

5.3.2 Phase 2: Project Management Phase (PM)

The Project Management category consists of processes which establish the project, and co-ordinate and manage its resources to produce a product or provide services which satisfy the customer.

The focus in this process category is on the effective use of resources (time, effort, people, money) toward accomplishing the purpose and objectives of the project.

The processes belonging to the Project Management Phase included:

PM.1 Establish Project Plan

PM.2 Establish Team & Training

PM.3 Manage Requirements

PM.4 Software Project Statement Scope Template

PM.1 Establish Project Plan

The purpose of the Establish project plan process is to establish reasonable plans for performing the software engineering and to form a basis for managing the software project.

Project plans typically document

- project purpose and objectives.
- broad timeline including project start,finish and major milestones.
- software budget estimates.
- perform risk assesment ,analysis and mittigation.
- schedule.
- resources allocated to the project activities(for eg software personnel ,hardware, software,services etc.).
- Identify special skills needed to accomplish project task.
- Recognise communication needs and develop communication Plan.
- Identify training needs and develop an appropriate strategy.

PM.1.1 Develop Work Breakdown Structure

Develop a work break down structure relating project tasks and sequence with the resources required to accomplish them.

PM.1.2 Identify Project Standards

Identify the software standards which will guide the project's software activities, consistent with the needs of the project.

PM.1.3 Identify Specialized Facilities

Identify any specialized tools, equipment, or rooms beyond those normally available which will be needed to meet any unusual technical requirements of the project.

Examples include

– target hardware;

– simulators;

– specialized test equipment;

PM.1.4 Identify Project Measures

Identify the basic set of status and other measures which will be used to track project progress and help determine if the project is meeting its objectives.

PM.1.5 Establish Project Schedule

Establish the project schedule, based on the software life cycle model, work breakdown structure, estimates, and risk mitigation plans.

PM.1.6 Establish Project Commitments

Establish commitments to the estimates and plans with all affected groups.

PM.1.7 Project Plan Template

Document the results of the activities in this process within the project plans.

PM.2 Establish Team & Training

The purpose of Build project teams process is to establish project teams with qualified members who can fulfil their responsibilities on their team and work together as a cohesive group.

The term "team" is intended to cover a number of different kinds and durations of teams, including

– mixed teams consisting of both customer and supplier representatives.

– review teams, development teams, problem analysis teams, process improvement teams etc.
– teams which exist for a short duration to solve a specific problem, or long-term as an established part of the environment.

PM.2.1 Define Project Teams

Define the teams which will be needed to perform the work of the project, defining the structure and operating rules for the team, required knowledge and skills.

PM.2.2 Empower Project Teams

Empower teams to perform their job, by ensuring that they have
– an understanding of their job
– a shared vision or sense of common interest
– appropriate mechanisms or facilities for communication and work
– support from the appropriate management for what they are trying to accomplish

PM.2.3 Maintain Project Team Interactions

Obtain and maintain agreement on the implementation of interactions between teams. Example areas on which to agree include responsibilites, interaction methods, conflict resolution methods.

PM.2.4 Manage Inter-Team Issues

Identify, track, and resolve issues that affect the progress of more than one team or threaten project unity.

The purpose of the training is to provide the organization and projects with individuals who possess the needed skills and knowledge to perform their roles effectively.

PM.3 Manage Requirements

The purpose of the Manage requirements process is to establish a software requirements baseline, which serves as the basis for the project's software work, products, and activities; and manage changes to that baseline.

PM.3.1 Agree on Requirements

Obtain agreement across teams on the customer's requirements, obtaining the appropriate sign-offs by representatives of all teams.

PM.3.2 Establish Customer Requirements Baseline

Document the customer's requirements and establish as a baseline for project use.

PM.3.3 Manage Customer Requirements changes

Manage all changes made to the customer requirements to ensure those who are affected by the changes are able to assess the impact and risks, and initiate appropriate change control and mitigation actions.

PM.3.4 Use Customer Requirements

Use the customer requirements as the basis for

– software project plans

– requirements specifications

– work products and activities

PM.3.5 Maintain Traceability

Establish and maintain traceability of requirements to the project's work products throughout the software life cycle.

PM.4 Software Project Scope Statement Template

Scope statements may take many forms depending on the type of project being implemented and the nature of the organization. Good scope definition is very important to project success because it helps improve the accuracy of time, cost and resource

estimates, it defines a baseline for performance measurement and project control, and it aides in communicating clear work responsibilities.

Project Name:	**Project Start Date:**
Project Manager:	**Date Prepared:**
Sponsor of the Project:	

Team Members:

Examples of who could be included in the Team Members section are:

- Note: as appropriate, include both internal (IT) and external team members.
- Can be any combination of individuals, groups, teams, or organizations.
- Include the core team members directly responsible for project deliverables.
- Optionally include supporting team members who contribute to project deliverables.

Purpose of Project:

Examples of what could be included in the Purpose of Project section are:

- Give a concise goal statement.
- What is being accomplished?
- What are the major benefits expected from this project?

Resource Requirements:

Examples of what could be included in the Resource Requirements section are:

- Benefit/Cost Analysis.
- Manage Resource & Schedule for achieving the Project objective & software requirement.
- Acquire Technical and Management resources
- Estimated total costs to include, e.g., major hardware/software purchases, ongoing production costs.
- Estimated data storage requirements and costs.
- Cross-organizational personnel requirements.

- Estimated training needs and costs.
- Broad timeline including project start, finish, and major milestones.

Safety, Security, & Risks.

Risk Level:

Examples of what could be included in the Safety, Security, & Risks section are (for example):

- Describe potential enhancements to personal safety and data security for the university, as a result of doing this project.
- Identify possible or known political or image exposures likely to result from this project.
- Identify possible or known IT security exposures for this project.
- Identify project risks for things like, acquisition, budget, personnel, contract, timeline, expectations, support, and training.
- Analyze & Prioritize Risks.
- For each identified issue or risk, identify at least one mitigation measure or a contingency plan.
- Take corrective action implementing mitigation strategies or contingency plan.

Risks	**Risk Mitigation or Contingency Plan**
1.	1.
2.	2.
3.	3.

Approvals by Key Officials:

Approved by: _____ **Date:** _____

Approved by: _____ **Date:** _____

Table 3: Project Scope Template

5.3.3 Phase 3: Engineering Phase (ENG)

The Engineering process category consists of processes that directly specify, implement, or maintain a system and software product and its user documentation.Engineering Phase mainly consists of impementation part.

Processes in Engineering Phase (ENG) are:

ENG.1 Develop Software Requirements

ENG.2 Develop Software Design

ENG.3 Implement Software Design

ENG.4 Integrate and Test system

ENG.5 Maintain System and Software

ENG.1 Develop software requirements

The purpose of the Develop software requirements process is to establish, analyze and refine the software requirements.

ENG.1.1 Determine Software Requirements

Determine the software requirements and document in a software requirements specification.

The software requirements specification describes such things as

– functions to be performed and their performance characteristics
– software interfaces (to hardware, operating system, and user) and their characteristics
– reliability characteristics
– installation and maintenance requirements
– safety requirements
– security characteristics

ENG.1.2 Analyze Software Requirements

Analyze the software requirements for correctness. Aspects of correctness to analyze incude completeness, understandability, testability, verifiability, feasibilty, validity and consistency

ENG.1.3 Evaluate Requirements with Customer

Communicate the software requirements to the customer, and revise if necessary, based on what is learned through this communication.

ENG.1.4 Update Requirements for next Iteration

After completing an iteration of requirements, design, code, and test, use the feedback obtained from use to modify the requirements for the next iteration.

ENG.2 Develop Software Design

The purpose of the Develop software design process is to establish a software design that effectively accommodates the software requirements; at the top-level this identifies the major software components and refines these into lower level software units which can be coded, compiled, and tested.

ENG.2.1 Develop Software Architectural Design

Transform the software requirements into a software architecture that describes the top-level structure and identifies its major components.

ENG.2.2 Design Interfaces at Top Level

Develop and document a top-level design for the external and internal interfaces.

ENG.2.3 Develop Detailed Design

Transform the top-level design into a detailed design for each software component. The software components are refined into lower levels containing software units that can be coded, compiled, and tested. The detailed design includes the specification of external and internal interfaces between the software units. The result of this base practice is a documented software design document which describes the position of each software unit in the software architecture and the functional, performance, and quality characteristics which each must address. These Parameters act as Quantified Feedback which can be further used in Business Process Optimizing Phase.

ENG 2.4 Establish Traceability

Establish traceability between the software requirements and software design.

ENG.3 Implement Software Design

The purpose of the implement software design is to produce executable and independently tested units of software code which implement the components of the software design.

ENG.3.1 Develop Software Units

Develop and document each software unit, including the code, data structures, data base. This base practice involves creating, documenting, and compiling representations of each software unit using expressions in the appropriate programming language.

ENG.3.2 Develop Unit Verification Procedures

Develop and document procedures for verifying that each software unit satisfies its design requirements. The normal verification procedure will be through unit testing, and the verification procedure will include unit test cases and unit test data.

ENG.3.3 Verify the Software Units

Verify that each software unit satisfies its design requirements and document the results.

ENG.4 Integrate and Test System

The purpose of the Integrate and test system process is to integrate the software with the manual operations and hardware elements producing a system that will satisfy the system requirements. This process is accomplished through developing aggregates of system elements and testing them as an aggregate, and then testing the resulting integrated system to ensure it satisfies the system requirements.

ENG.5 Maintain System and Software

The purpose of the Maintain system and software process is to modify the system, its hardware, the network system, software, and associated documentation in response to user requests while preserving the integrity of the system design.

There are several sources which create the need for modifying the system or software. Example sources include

– detected error

– deficiency

– problem in the operation of the system or software

– particular improvement or modification of the system or software required or requested by the customer.

ENG.5.1 Determine Maintenance Requirements

Determine the system and software maintenance requirements, identifying the system and software elements to be maintained, and their required enhancements.

ENG.5.2 Analyze User Problems and Enhancements

Analyze user problems and requests and required enhancements, evaluating the possible impact of different options for modifying the operational system and software, system interfaces, and requirements.

ENG.5.3 Determine Modifications for next Upgrade

Based on the above analyses, determine which modifications should be applied in the next system or software upgrade, documenting which software units and other system elements and which documentation will need to be changed .

ENG.5.4 Implement Modifications

Use the other engineering processes, as appropriate, to implement and test the selected modifications, demonstrating that the unmodified system and software requirements will not be compromised by the upgrade.

ENG.5.5 Train Customer

Provide training and documentation, as appropriate, to the customer so that the software can be effectively used.

5.3.4 Phase 4: Business Process Optimizing Phase

The processes belonging to Business Process Optimizing Phase are:

BPO.1 Improve Standard Software Processes

BPO.2 Improving Human Aspects

BPO.3 Put in Action the Improved Processes effectively

BPO.4 Cost Benefit Analysis

BPO.5 Preventing Defects

BPO.6 Quantitative Feedback

BPO.1 Improve Standard Software Processes

The purpose of improving standard processes is to enhance the standard processes which are followed by the organization. In short, Problem with the Process rather than the technology cause a number of difficulties. Once a standard process is in place, it can be improved every time a new project is completed. This type of corporate experience will contribute directly to the bottom line in terms of reduced cost and increased business.

BPO.1.1 Implement Advance Technology and Innovations

Piloting innovative Ideas and new technologies for enhance the standard of existing software processes in the Organization This type of change in the standard software process also support the Continuous Process Improvement.

BPO.1.2 Use of Open Source Tools and Methodologies

Use open source tools and methodologies for software which are easily available and cost effective. New and better tools and methodologies will continue to be produced, just as new equipment. A standard process will provide a framework for controlled change, allowing maximum use of better tools and methods.

BPO.1.3 Commitment to Learning

Continuous improvement also requires a commitment to learning on the part of the organization. Commitment to learning is Important for ‘‘learning organizations”, i.e. systematic problem solving, experimentation with new approaches, learning from their own experience and past history, learning from the experiences and best practices of others, and transferring knowledge quickly and efficiently throughout the organization.

BPO.2 Improving Human Aspects

Improve the Human Aspects because the manpower motivation if increased, it directly increases the improvisation of process.

Improvising human aspects will result in process Improvisation. only improving processes and ignoring human aspects will affect the performance of the process. So proper strategies to motivate personnel are will directly contribute to continuous improvement of manpower and processes which will indirectly increase the customer satisfaction.

BPO.3 Put in Action the Improved Processes

The Improved Processes that exploit the best software engineering practices are put in action & transferred throughout the organization.

BPO.4 Cost Benefit Analysis

Cost-benefit analysis (CBA) will be a good way because it visualizes what they can gain and lose in acquiring the new technologies and proposed changes to the organization software process. Implementation cost means the overhead cost in adopting the new knowledge and management skills delivered. Implementation cost can be quantified by the opportunity cost about the lost productivity between the productivity before a legacy process and that after a new process.

BPO.5 Preventing Defects

Defect prevention is very much vital for the growth of an organization’s quality levels. The main intention of the thrust on quality is not to reduce the cost but to invest the cost

for right returns. It saves a lot of rework that are required when the defect gets manifested at the final stages or at the post delivery period. Defect prevention should be introduced at every stage of the software life cycle to block the defects at the earliest. If effective DP process can be functional at all stages of software development, then the creation of near zero defect products is conceivable.

BPO.5.1 Pro-active Defect Prevention

Pro-active DP focuses on creating an environment to control defects rather than one that simply responds to them. A kick off meeting is conducted for defect prevention prior to the start of each life cycle phase or task to outline the areas where mistakes were committed and actions taken for their rectification. The company deems from the previous projects, lessons that were learnt from the life cycle phases, DP action items documented and the best practices adopted.

BPO.5.2 Reactive Defect Prevention

The reactive DP identifies and conducts Root Cause Analysis (RCA) for the defects. Causal analysis is a sporadic assessment to identify the root causes of the defects by means of the methods like cause/effect diagrams, Pareto analysis etc. The corresponding corrective actions are implemented along with preventive actions to eradicate future defects. The most common root causes for defects identified are due to miscommunication, lack of training, oversight, project methodology and planning.

BPO.5.3 Retrospective Defect Prevention

Retrospection is performed towards the end of the project or at identified phases to identify strong points and to explore the areas requiring perfection.

BPO.6 Quantitative Feedback

Quantitative Feedback data from the process allows continuous process improvement. Data gathering has been partially automated as the data and documentation from each phase is collected in the supporting set. Panel of experts from inside the organization can

be established. The experts can refer the quantitative feedback and also the related documentation from the supporting set and can make a GO or noGO decision for the next iteration of SPIMS to carry out. Performance, functional quality characteristics evaluated in detailed design in ENG. 2.3.

5.3.5 Supporting Set

The Supporting set consists of activities which may be employed by any of the other processes.The supporting activities can be employed at various life-cycle and can be implemented with different Phases of Process models.

The activities belonging to the Supporting set are

SS.1 Develop Documentation

SS.2 Perform Configuration Management

SS.3 Perform Problem Resolution

SS.4 Perform Peer Reviews

SS.5 Software Reuse (Optional)

SS.1 Develop documentation

The purpose of the Develop documentation process is to develop and maintain documents needed by managers, engineers, users, or customers of the system or software.

This process covers the development of documents such as

– Project management documentation, such as plans.

– Engineering work product documentation, such as design rationale.

– Process documentation, such as a peer review procedure.

– and end user documentation, which describes the intended use of the system and software to a user.

SS.2 Perform Configuration Management

Software Configuration Management is a set of activities that have been develop to manage and control changes throughout the entire software life cycle. It is responsible to identify software configuration items, control various versions, control changes and report status.

SS.2.1 Establish Configuration Management Library System

Establish and manage a repository with access controls that provides for

– storage and retrieval of configuration items (and their versions)

– sharing and transfer of configuration items between affected groups

– recovery of archive versions of configuration items

SS.2.2 Manage Change Requests

Record, review, approve, and track all change requests and problem reports for all configuration items and their versions.

SS.2.3 Control Changes

Provide access control to help maintain the correctness and integrity of the software items in the configuration management library system.

SS.2.4 Build Product Releases

Build product releases only from configuration items in the library and only when authorized.

SS.2.5 Maintain Configuration Item History

Maintain a history of each configuration item, recording configuration management actions against the item in sufficient detail to allow for recovery of previous versions.

SS.2.6 Report Configuration Status

Regularly report on the results of performing the above activities and the status of each configuration item.

SS.3 Perform Problem Resolution

The purpose of problem resolution process is to ensure that all discovered problems are analyzed and removed, and trends are identified.

The problems being addressed here are any detected problems whatever their nature or source. Example sources are base practices

– Problems identified during software testing in ENG. 4.0.
– User problems identified during maintenance in ENG 5.2.
– Problems identified during peer reviews in SS. 5.5.

There are many other sources among the base practices, for example, those involving taking corrective action.

SS.4.1 Prepare Problem Report

Prepare a problem report promptly after each problem is detected describing the nature of the problem. It may be the user or customer who does this, in which case this base practice can be considered.

SS.4.2 Track Problem Report

Track the resolution of the problem/change report.

SS.4.3 Prioritize Problems

Categorize and prioritize according to the priority and category of the problem.

SS.4.4 Determine Resolution

Analyze the problem and, if possible, determine the problem's cause and document its resolution.

SS.4.5 Correct the Defect

Eliminate the defect in the product.

SS.4.6 Distribute the Correction

Distribute the corrected product.

SS.5 Perform Peer reviews

The purpose of the Perform peer reviews process is to efficiently find and remove defects from products produced by the project. It is important to hold a review early in the task so that defects can be efficiently found and rework significantly reduced. In such reviews, the emphasis is on technical correctness, and so the reviewers are typically the colleagues of the work product's producer.

SS.5.1 Select Work Products

Identify the work products that are to undergo peer review.

SS.5.2 Identify Review Standards

Identify the standards (including checklists) to be used in conducting the peer reviews.

SS.5.3 Establish Completion Criteria

Establish the completion criteria for successful completion of peer reviews.

SS.5.4 Establish Re-review Criteria

Establish criteria for when and how to re-review a work product.

SS.5.5 Distribute Review Materials

Distribute the materials for peer reviews well in advance of the reviews.

SS.5.6 Conduct Peer Review

Conduct peer review on the selected work product.

SS.5.7 Document Action Items

Document action items identified during peer reviews.

SS.5 Enable Reuse

This is an optional activity in Supporting set. The purpose of the Enable reuse process is to maximize reuse of existing system and software components, leading to lower development and maintenance costs and higher quality product lines.

SS.5.1 Identify Reusable Components

Identify system and software components which could profitably be reused within the product lines established by the organization. Consider only high quality system and software components for reuse.

SS.5.2 Develop Reusable Components

Develop system and software components which are designed for reuse.

SS.5.3 Establish a Reuse Library

Establish a library of reusable components, including the mechanisms for identifying and retrieving components.

SS.5.4 Certify Reusable Components

Certify that components placed in the reuse library have been appropriately packaged for reuse.

Chapter 6

Conclusion & Future Work

Conclusion

SPIMS Framework provides an **iterative, specialized cum generalized process model** which encompasses small and effective set of processes. Small Organizations are not capable of bearing the cost of establishing software process improvement programs and the existing software process standards or models are not easily applicable. Thus, a SPIMS model is proposed.

Iterative Model

The proposed SPIMS model is an iterative model. The basic idea behind is to make SPIMS iterative to enhance and develop a software system incrementally. SPIMS can iterate on previous requirements depending upon the GO/noGo decision given by experts. if the product quality is not up to mark and can also increment in the next iteration if and only if there are some new requirements to be fulfilled.

Specialized Model

SPIMS incorporates some specialized techniques introduced in its different phases. In order to follow any process improvement model, the main emphasis of small scale industries is on reducing cost, time and enhances product quality. SPIMS Model differs from other models as SPICE, BOOTSTRAP, PSP gives the wide variety of processes which are very generalized in nature. The Small scale industries find the process models confusing in selecting some particular processes from a large generalized set of processes. SPIMS's specialized techniques can be implemented with any kind of projects therefore making small scale industries more clear about their goals and strategy to follow. These specialized techniques in SPIMS help to reduce cost, time and thereby improve product quality.

Generalized Model

SPIMS Model have a different set of supporting activities which encompasses the general techniques. These supporting activities can be chosen by the small scale industry depending upon their project's objectives. The idea behind the supporting set is that the activities in supporting set can support any phase with industries discreation. Supporting activities in supporting set can be implemented with different phases of process model depending upon the industry's needs and requirements, it can select any activity from supporting set and apply it with any phase. So, SPIMS also have generalized processes in supporting set which can be selected and exercised by small scale industry.

Small scale industries aim at a process model which can provide small and effective set of processes those can fit into their time and cost constraints and also is capable to improve quality and customer satisfaction. SPIMS provides a practical and comprehensive framework that gives small scale organizations the opportunity to have an improved software product quality.

Future work

Future work would involve empirical evaluation of this process model and identifying appropriate metrics for such evaluation. Software metrics are being identified to provide the specific information necessary to manage software projects and improve software engineering processes and services. Measuring software is a powerful way to track progress towards project goals. Without such measures for managing software, it is difficult for any organization to understand whether it is successful. Appropriately identified metrics can help both management and engineers maintain their focus on their goals.

References

[1] Werth, Laurie Honour**, "**Introduction to Software Process Improvement", Carnegie Mellon University, CMU/SEI-93-EM-8.

[2] Zahran, Sami, "Software Process Improvement Practical Guidelines for Business Success", Addison Wesley Longman, 1998.

[3] Humphrey, Watts S, "Introduction to the Personal Software Process", Addison Wesley Publishing company, 1996.

[4] Salo, Outi, "Enabling Software Process Improvement in Agile Software Development Teams and Organizations". Espoo 2006.

[5] Humphrey, W. S, "Managing the Software Process", Addison-Wesley, Reading, MA, 1989(B).

[6] Iversen, Jakob, Ngwenyama, Ojelanki."Problems in measuring effectiveness in software process improvement: A longitudinal study of organizational change at Danske Data', ELSEVIER, 2006

[7] Villalón, jose a., calvo-manzano, Agustín, gonzalo cuevas., Gilabert, tomás san feliu., "Experiences in the Application of Software Process Improvement in SMES", Software Quality Journal, 10, 261–273, Kluwer Academic Publishers, 2002

[8] K. Butler, "The Economic Benefits of Software Process Improvement," cross talk, July 1995, pp. 14-17

[9] V. Basili, and S. Green, "Software Process Evolution at the SEL," IEEE Software, V.11, N.4, July 1994, pp.58-66

[10] McFeeley, IDEAL SM: "A User's Guide for Software Process Improvement. Handbook" CMU/SEI- 96-HB-001. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PE, USA.

[11] Humphrey W.S, "Introduction to the Personal Software Process", ISBN 0-201-54809-7, Addison-Wesley, 1997

[12] Humphrey, W.S, **"**Using a defined and measured personal software process", IEEE Software, May 1996

[13] Humphrey, W.S, "A personal commitment to software quality", in: American Programmer, December 1994

[14] Dorling A, **"**SPICE Software Process Improvement and Capability determination", Software Quality Journal 2, 209-224 (1993)

[15] Rout, Terence P, SPICE: "A Framework for Software Process Assessment", Software Process - Improvement and Practice, Pilot Issue, pp 57-66 (1995)

[16] K. El Eman, J. Drouin and W. Melo, "SPICE The Theory and Practice of Software Process Improvement and Capability Determination", IEEE Computer Society, 1997, ISBN 0-8186-7798-8

[17] M.C. Paulk, C.V. Weber, S. Garcia, M.B. Chrissis, and M. Bush, "Key Practices of the Capability Maturity Model", Version 1.1, Software Engineering Institute, CMU/SEI-93-TR-25, February 1993.

[18] C.V. Weber, M.C. Paulk, C.J. Wise, and J.V. Withey, "Key Practices of the Capability Maturity Model", Software Engineering Institute, CMU/SEI-91-TR-25, ADA240604, August 1991.

[19] Paulk, M.C, "The Capability Maturity Model: Guidelines for Improving the Software Process", ISBN 0-201-54664-7, Addison-Wesley Publishing Company, 1994

[20] Haase, W. R. Messnarz, G. Koch, H.J. Kugler and P. Decrinis**, "**BOOTSTRAP: Fine tuning process assessment", IEEE Software, pp. 25-37, July 1994

[21] BOOTSTRAP team (1993), "BOOTSTRAP Europe's assessment method", IEEE Software, pp. 32-35, May 1993

[22] Kuvaja, P., J. Simila, L. Krzanik, A. Bicego, G. Koch and S. Saukonen, "Software Process Assessment and Improvement: the BOOTSTRAP approach" Blackwell Publishers, Oxford, UK, 1994

[23] Dyba, T. 2000, "An Instrument for measuring the key factors of success in SPI", Proceeding of IEEE, pp.357-390, February 1990.

[24] Hall, T. and Wilson, D. 1997. "Views of software quality: a field report", IEEE Proceedings on Software Engineering 144 (2).

[25] Stelzer, D. and Werner,"Success factors of organizational change in software process improvement", Proceedings of IEEE, pp.76-85, July 1998.

[26] Wilson, D. and Hall, "Perceptions of software quality: a pilot study", Software quality journal (7). 67-75.

[27] Debou, Christophe., Combelles, Annie Kuntzmann., "Linking Software Process Improvement to Business Strategies: Experiences from Industry", software process improvement and practice 2000, John Wiley & Sons, Ltd., 2000

[28] Brahman, Judith G. & Johnson, Donna L. (1994), "What small business and small organizations say about the CMM: experience report", Proceedings of the 16th international conference on Software engineering, pp-331-340, Sorrento, Italy

[29] Cattaneo, Fabiano; Fuggetta, Alfonso; & Lavazza, Luigi. (1995) "An experience in process assessment", Proceedings of the 17th international conference on Software engineering, pp-115-121, Seattle, Washington

[30] Dyba, Tore, "Factors of software process improvement success in small and large organizations" ACM SIGSOFT Software Engineering Notes. 28(5).

[31] Pino, Francisco J.; García, Felix; & Piattini, Mario, "Software process improvement in small and medium software enterprises: a systematic review", Software Quality Control. 16(2). pp.237-261, September 1985.

[32] Software Engineering Institute, "The IDEAL Model" Retrieved October 10, 2008 from the World Wide Web: http://www.sei.cmu.edu/ideal

[33] Saiedian, Hossein & Carr, Natsu, "Characterizing a software process maturity model for small organizations", ACM SIGICE Bulletin. 23(1). 211.

[34] Sutton, Stanley M, "The Role of Process in a Software Startup", IEEE Software. 17(4). Pp.33-39, July 1986.

[35] Emam, Khaled El & Birk,"Validating the ISO/IEC 15504 Measure of Software Requirements Analysis Process Capability", IEEE Transactions on Software Engineering. 26(6). pp.541-566, August 1995.

List of Papers

[1] Dinesh Tagra, Ms. Ashima Singh **"Why Small Scale Industries should adopt Software Process Improvement Models"** at International Conference on "Information System & Software Engineering", ICISSE-2009, Meenakshi Sundarrarajan Engineering College, Chennai, India. (Status: Submitted)