SOFTWARE PROCESS CUSTOMIZATION AND IMPROVEMENT TOOL

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

Master of Engineering
in
Software Engineering

Submitted By:
HARSH TANEJA
(801031010)

Under the supervision of:
MS. ASHIMA SINGH
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2012

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "**Software Process Customization and Improvement Tool**", in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering 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 work which are duly listed in the reference section.

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

Harsh Taneja

(Harsh Taneja)

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

Ashima Singh

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

Countersigned by:

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Thesis Acknowledgement

Foremost, I would like to express my sincere gratitude to my guide Ms. Ashima Singh for the continuous support of my M.E. study and research, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.

Besides my guide, I would like to thank the rest of the faculty of Computer Science and Engineering Department, Thapar University, Patiala for their encouragement, insightful comments and support towards my work .

I thank my fellow batch mates in Thapar University for the discussions, for the hard work, and for all the fun we have had in the last two years. Also I thank my friends residing outside the university for their help and support in my thesis work.

Last but not the least, I would like to thank my family for supporting me spiritually & making me mentally strong throughout my life to extract maximum out of me.

Harsh Taneja

Harsh Taneja

Abstract

Software Process Improvement (SPI) has now become a milestone. Every small, medium or large organization want to achieve it. There are many organizations those want to improve the processes but in their own way. Therefore SPI is now moving towards Process Customization, that is they want to select processes with respect to their budget, time and resources.

Different advances have been made in the development of software process improvement (SPI) standards and models, e.g. capability maturity model (CMM), CMMI, and ISO's SPICE. However, these advances have not been matched by equal advances in the adoption of these standards and models in software development which has resulted in limited success for many SPI efforts. The current problem with SPI is not a lack of standard or model, but rather a lack of an effective strategy to successfully implement these standards or models. The importance of SPI implementation demands that it be recognized as a complex process in its own right and that organizations should determine their SPI implementation maturity through an organized set of activities and process areas. In the thesis, much attention has been paid to ''what activities to implement'' instead of ''how to implement these activities".

The Software Process Customization and Improvement (SPCI) tool is developed in order to cater the needs of Software Industry w.r.t their available resources. Process and related activities are identified from various available models and grouped into various categories to make customized software available to organizations. Now, they can select the proper set of activities according to their need and budget from the tool developed and can proceed to next level with maturity. Thus Software Development Process being followed is improved to enhance the quality of product time to time.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Software Development Strategies

Various development methods and processes are employed for software today. Prime focus is on designing, creating, and maintaining large-scale application software developed by software development teams. Developing huge operating systems with millions of lines of code (LOC)/Function Points(FP), or large, complex systems such as Indian Railway Reservation System needs a planned and procedural way to develop it [20]. The software development process generally follows these steps:-

i. Specification or functional design, done by system analysts in consent with the potential end users of the software to determine *why* to do this, *what* the application will do, and *for whom* it will do it.
ii. Architecture or technical design, done by system designers as the way to achieve the goals of the functional design using the resources available, This is *how* the system will function.
iii. Programming or implementation, done by computer programmers together with the system designers.
iv. Testing of the systems to ensure that the goals of the functional design and technical design are met.
v. Documentation of the system, both intrinsically for its future maintainers, and extrinsically for its future users.
vi. Maintenance of the application system post-development, employing the design document now used as the Technical Specification or System Maintenance Document.

We can have a lots of software development process models with improved cost and schedule management and will be getting a countless numbers of such models because of research activities being carried out in this area as covered in “Literature Survey”. But we will always need a general environment for software development and its improvement as shown in figure 1.1.

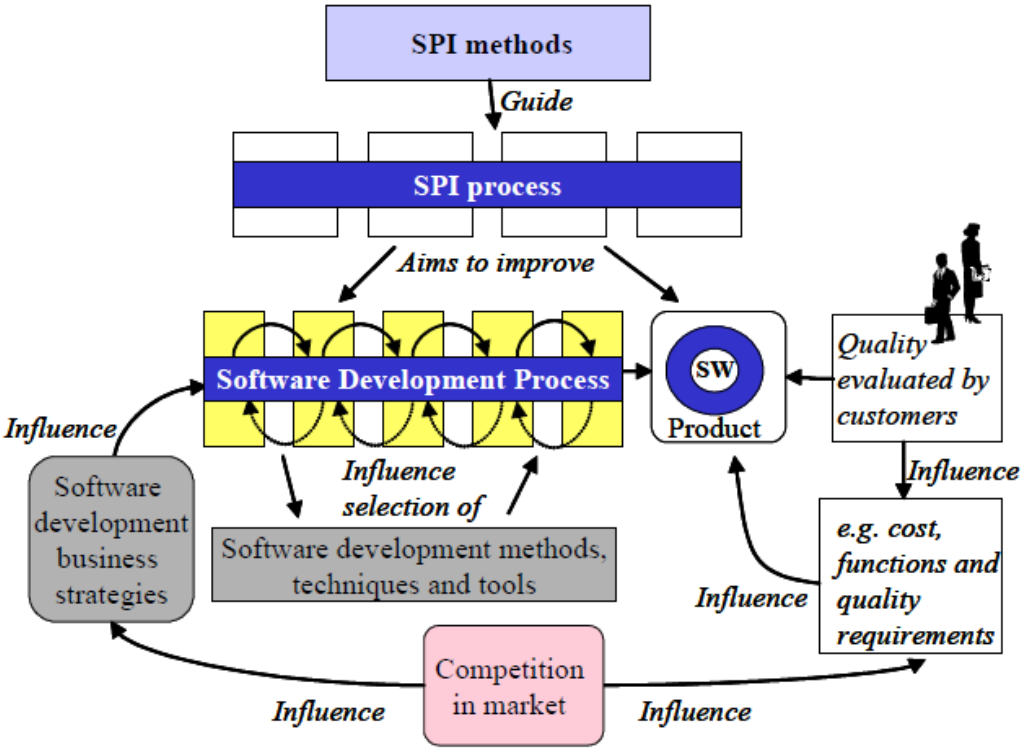

Figure 1.1: Environment for software development [31]

1.2 Software Development Process Customization and its Improvement

Some of the software development processes are so much complicated that they can be used leaving certain process areas from a fixed software development model without affecting the quality of software being developed using that software development process and sometimes the need to use the process area of another software development model if it is compatible with the software development model being used. Customizing the software development process is helpful for improving the development process by gaining maturity with time.

Here we have done customization for process improvement based on the process areas and corresponding activities needed for development of a project and the capabilities of the organization developing the project. The focus is also on improving the software development process so that quality of the product being developed using new process is improved one.

As an organization matures, the software process becomes better defined and more consistently implemented throughout the organization. Software process capability describes the range of expected results that can be achieved by following a software process [9]. The

software process capability of an organization provides one means of predicting the most likely outcomes to be expected from the next software project the organization undertakes. Continuous effort is done to improve the process of development of software. One of those efforts can be seen in form of PDCA method as shown below:-

1.2.1 PDCA method

Plan-Do-Check-Act (PDCA) is a continuous improvement cycle developed by Walter Shewhart at Western Electric and popularized by Dr. W. Edwards Deming [39]. It consists of four phases –
plan, do, check and act. **Planning** involves deciding what data is needed and how it will be collected. This phase requires an analysis and selection of alternative improvements. **Do** consists of carrying out the planned change. **Checking** assesses the results of the change. **Act** places the most effective alternative as the standard mode of operation [33]. Then, the cycle starts again with a new set of planned improvements.

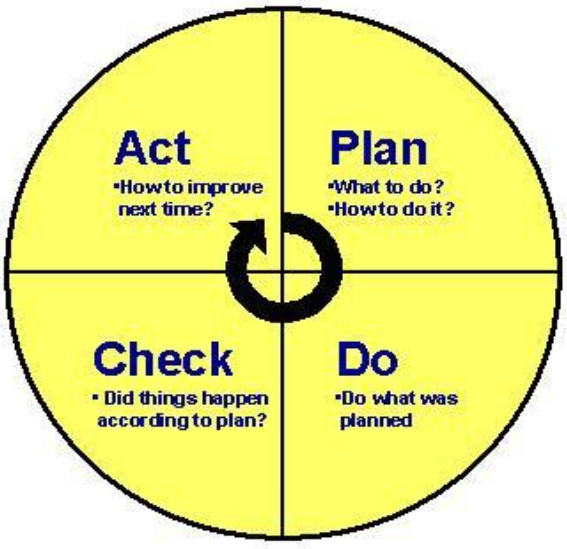

Figure 1.2: THE Plan-Do-Check-Act (PDCA) Cycle [39]

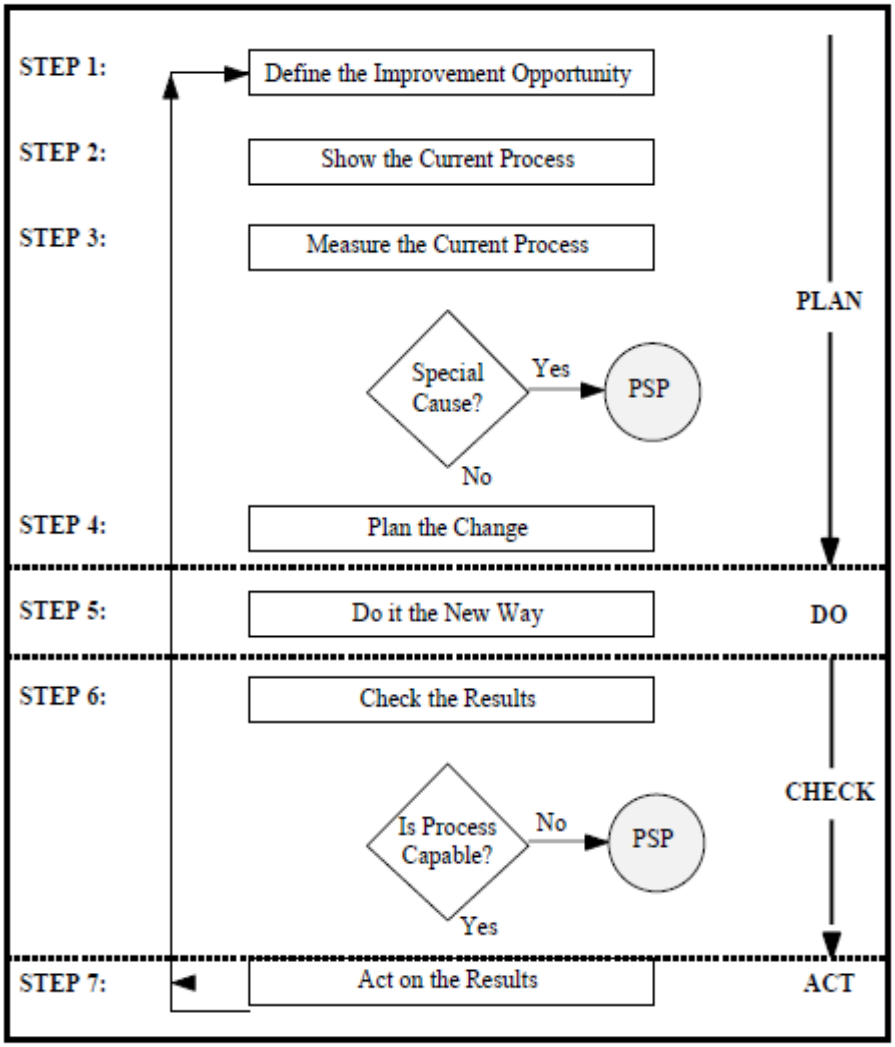

Figure 1.3: Flowchart of PDCA [39]

To conclude, we may have a lot of Software Development models available with us as discussed in Literature Survey to be followed, but a general environment for software development and its improvement will always be needed to generate a new model as a base for further improvement of Software Development Process.

Chapter 2

Literature Survey

2.1 Basic Software Development Metodologies

A software development model is an organized procedure for carrying out the steps in the life cycle of a software application program or system in a predictable, reliable and efficient way [13]. Here we will begin with the perspective models, of which there are many variants. These are the build-and-fix model, the waterfall model, the evolutionary model, the spiral model, and many more. It is not our purpose to compare existing software development technology in any detail. We only want to establish a general context and their introduction for getting knowledge about our new approach [25].

2.1.1 Build-and-Fix Model

The *build-and-fix* model was adopted from an earlier and simpler age of hardware product development. It was used in Volkswagen automobiles in the 1950s and ’60s. As new models were brought out and old models updated, the cars were sold apparently without benefit of testing, only to be tested by the customer. In every case, the vehicles were repaired by the dealer at no cost to their owners, except for the inconvenience and risk of a breakdown. Unfortunately in the build-and-fix approach, the product’s overall quality is never really addressed, even though some of the development issues are ultimately corrected. Corrections are put back into the market as bug fixes, service packs, or upgrades as soon as possible. Thus, little learning takes place within the development process. Because of this, build-and-fix is totally reactive and, by today’s standards, is not really a development model at all.

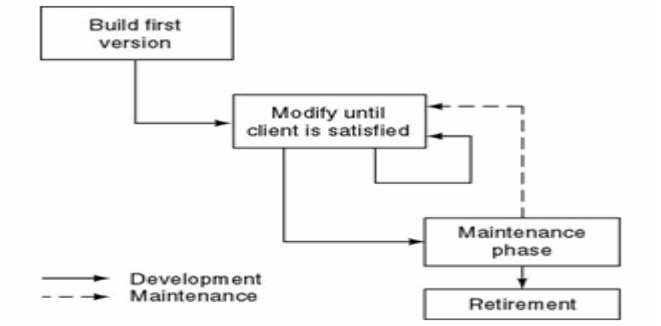

Figure 2.1: Build-and-Fix Software Development Model [34]

2.1.2 Waterfall Model

The classic waterfall model was introduced in the 1970s by Win Royce at Lockheed. It is so named because it can be graphically represented as a cascade from establishing requirements, to design creation, to program implementation, to system test, to release to customer. It was a great step forward in software development as an engineering discipline. The figure also depicts the single-level feedback paths that were not part of the original model but that have been added to all subsequent improvements of modified waterfall model. The original waterfall model had little or no feedback between stages, just as water does not reverse or flow uphill in a cascade but is drawn ever downward. This method might work satisfactorily if design requirements could be perfectly addressed before flowing down to design creation, and if the design were perfect when program implementation began, and if the code were perfect before testing began, and if testing guaranteed that no bugs remained in the code before the users applied it, and of course if the users never changed their minds about requirements and requirements have been freeze beforehand. Some simple hardware products may be designed and manufactured this way, but this model has been unsatisfactory for software products because of the complexity issue [18], [38]. Thus the drawback of waterfall model is that it is simply impossible to guarantee correctness of any program and alteration of any requirement.

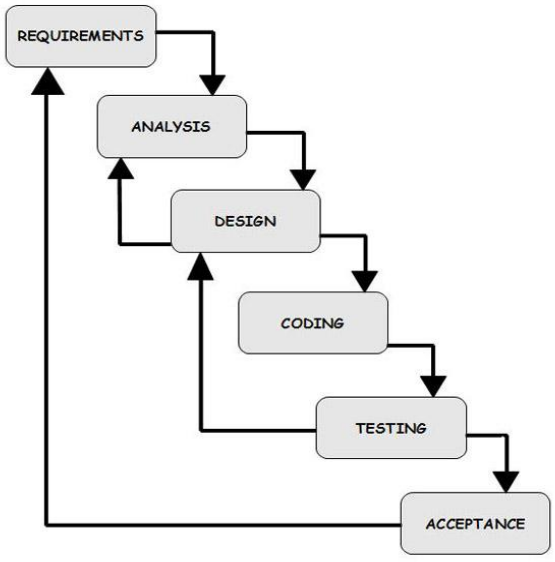

Figure2.2: Waterfall Model for Software Development [18]

2.1.3 Incremental Model

The incremental model recognizes that software development steps are not discrete. Instead, Build 0 (a prototype) is improved and functionality is added until it becomes Build 1, which becomes Build 2, and so on. These builds are not the versions released to the public but are merely staged compilations of the developing system at a new level of functionality or completeness. It is intended to deliver an operational-quality system at each build stage, but it does not yet complete the functional specification. One of the biggest advantages of the incremental model is that it is flexible enough to respond to critical specification changes as development progresses. Another clear advantage is that analysts and developers can tackle smaller chunks of complexity [34]. Users and developers both learn from a new system's development process, and any model that allows them to incorporate this learning into the product is advantageous. The drawback is, of course, that learning exceeds productivity and the development project becomes a research project exceeding time and budget or, worse, never delivers the product at all. Thus, learning need not exceed productivity if the development team remains focused on customer requirements.

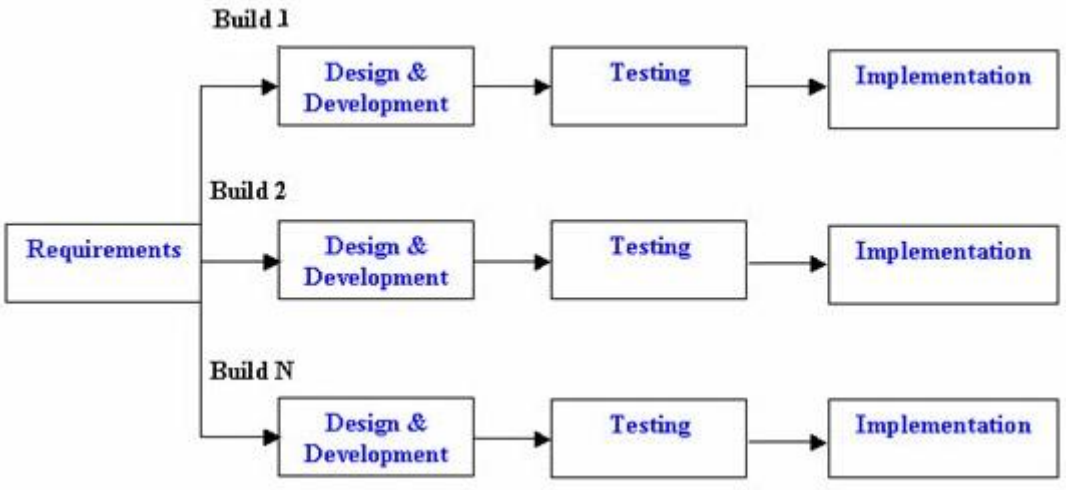

Figure 2.3: Incremental model for software development [34]

2.1.4 Spiral Model

The spiral model, developed by Dr. Barry Boehm at TRW, is an enhancement of the waterfall/rapid prototype model, with risk analysis preceding each phase of the cascade. This model has been successfully used for the internal development of large systems and is

especially useful when software reuse is a goal and when specific quality objectives can be incorporated. It does depend on being able to accurately assess risks during development [7]. This depends on controlling all factors and eliminating or at least minimizing risk factors. Like the other extensions of and improvements to the waterfall model, it adds feedback to earlier stages. This model has been used in the development of major programming projects over a number of years.

2.1.5 CMM

The Capability Maturity Model for Software provides software organizations with guidance on how to gain control of their processes for developing and maintaining software and how to evolve toward excellence. The CMM was designed to guide software organizations in selecting process improvement strategies by determining current process maturity and identifying the few issues most critical to software quality and process improvement. By focusing on a limited set of activities and working to achieve them, an organization can steadily improve its organization-wide software process to enable continuous and lasting gains in software process capability [1]. These principles have been adapted by the SEI into a maturity framework that establishes a project management and engineering foundation for quantitative control of the software process, which is the basis for continuous process improvement. Continuous process improvement is based on many small, evolutionary steps. The CMM provides a framework for organizing these evolutionary steps into five maturity levels that lay successive foundations for continuous process improvement [23], [16]. These five maturity levels define a scale for measuring the maturity of an organization's software process and for evaluating its software process capability. The levels also help an organization prioritize its improvement efforts as follows:-

i. **Initial:** The software process is characterized as ad hoc. Few processes are defined, and success depends mostly on individual effort.

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 innovative ideas and technologies [30].

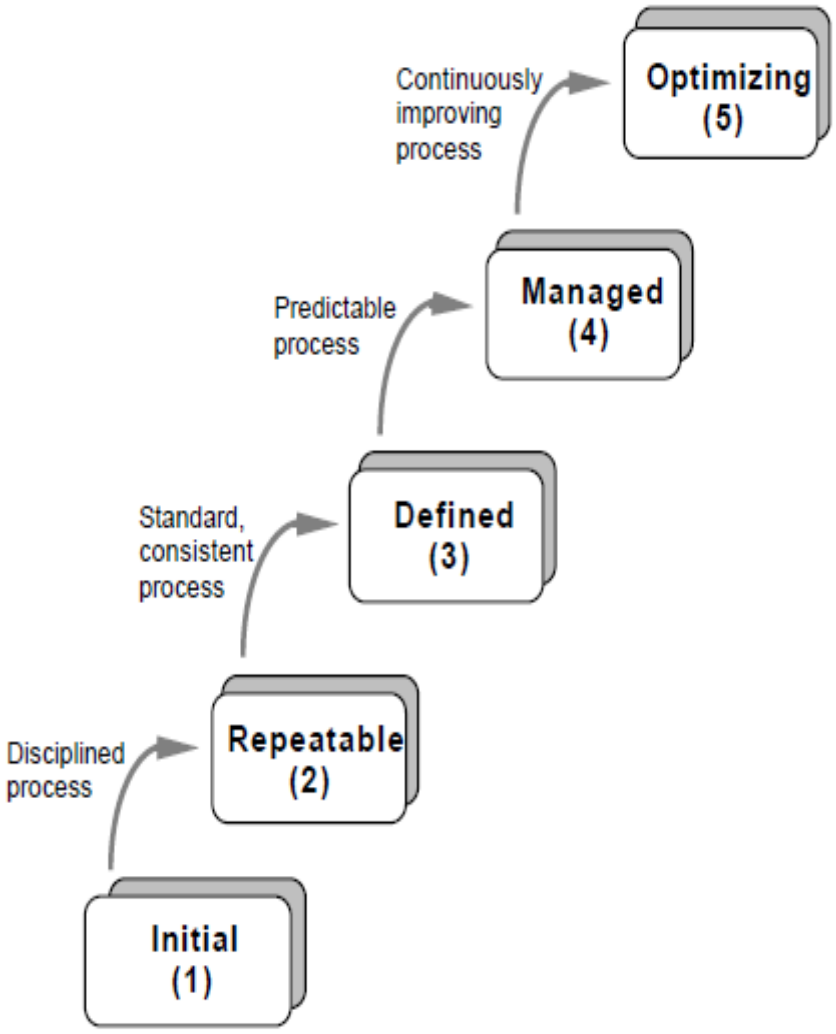

Figure 2.4: Five levels of Software Process Maturity in CMM [23]

2.1.6 BOOTSTRAP

The Bootstrap method evaluated from a European Community project (ESPRIT project 5441). The methodology was designed to determine the maturity of an organization, its strengths and weaknesses and the improvement guidelines. Since 1993 the project has been managed by the Bootstrap Institute. In 1997 the Bootstrap version 3.0 was released. This version has been developed to be fully compliant with ISO/IEC 15504 [3].

The objectives of the Bootstrap methodology are:-

i. Provide support to the evaluation of process capability against a set of recognised software engineering best practices.
ii. Include internationally recognized software engineering standards as sources for identification of best practices.
iii. Support the evaluation of how the reference standards have been implemented in the assessed organization.
iv. Identify strengths and weaknesses in the organization's processes.
v. Provide benchmarking data for assessment of results.
vi. Provide results that form a suitable and reliable basis for improvement planning.
vii. Plan improvement actions that support achievement of the organization's goal

The Bootstrap is presented as an assessment model including three dimensions: a process dimension, a capability dimension and a technology support dimension. The process dimension is called the bootstrap process model. It integrates requirements from several standards like ISO9001 (1989), ISO9000-3(1991), ISO12207 (1995), ISO15504 (1993), ESA PSS-05-0 (1991) and the CMM [2], [35]. The Bootstrap process architecture reflects a tree structure as shown in figure 2.5 that contains: process categories, processes and best practices.

Figure2.5: Bootstrap model [3]

2.1.7 CMMI

Since the first version in 1991, several CMM's have been developed for different disciplines which include models for systems engineering, software engineering, software acquisition, workforce management and development, and integrated product and process development. These models were very useful for the organizations but multiple uses inside an organization became problematic. Many organizations would like to focus their improvement efforts across the disciplines in their organizations. However, the differences among these disciplines-specific models have limited these organizations ability to focus their improvements successfully [15], [21]. The use of multiple models in an organization is also costly in terms of trainings and improvement activities. There are three main sources models used in CMMI development:-

i. Software: SW-CMM, draft version 2(C)
ii. System engineering: EIA/IS 731
iii. Integrated product and process development: IPD CMM, version 0.98

These models were selected because of their widespread adoption in the software and systems engineering and because of their different approaches to improve processes in an organization. There are multiple CMMI models available, as generated from the CMMI framework. There are two representations available: *Continuous* and *Staged* [3]. Both representation presents their own characteristics and the organization must chose the representation according to its objectives.

Continuous representation:-

i. Allow you to select the order of improvement that best meets the organization's business objectives and mitigates the organization's area of risk
ii. Enable comparisons across and among organizations on a process area by process area basis or by comparing results through the use of equivalent staging
iii. Provide an easy migration from EIA/IS 731 to CMMI
iv. Afford an easy comparison of process improvement to ISO/IEC 15504

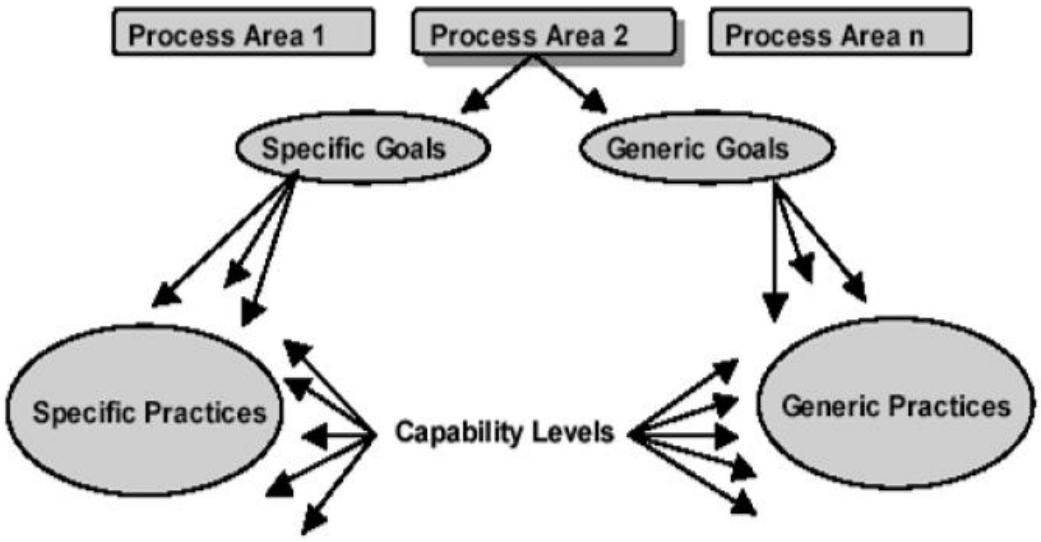

Figure2.6: Continuous Representation [3]

Staged representation:-

i. Provide a proven sequence of improvements, beginning with basic management practices and progressing through a predefined and proven path of successive levels.
ii. Permit comparisons across and among organizations by the use of maturity levels.
iii. Provide an easy migration from SW-CMM to CMMI.
iv. Provide single rating that allows comparisons among organizations.

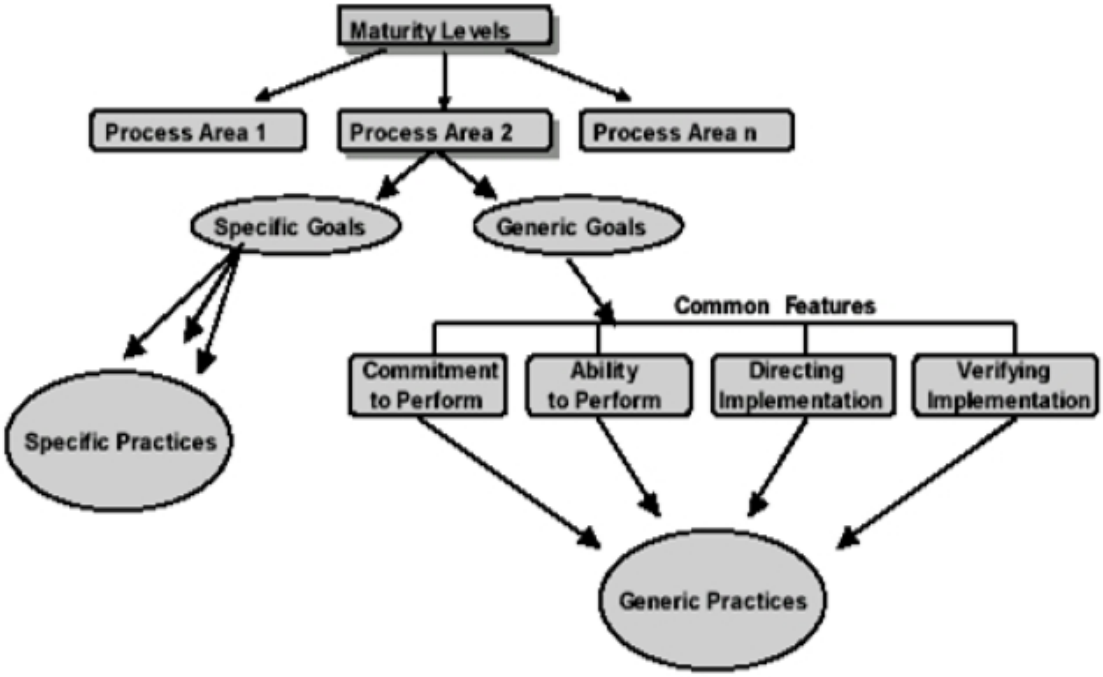

Figure 2.7: Staged Representation [3]

2.2 Current Trends and Approaches to Software Process Customization:-

2.2.1 Wipro's Idea of Customization: Veloci-Q

Six Sigma methodologies were used to enable higher levels of optimization, resulting in lower costs and increased customer satisfaction. In fact, Six Sigma directly contributes to CMM Level 5—**continuous optimization of key process areas**. Six Sigma achieves dramatic improvement in business performance through a precise understanding of customer requirements and the elimination of defects from existing processes, products, and services.

Institutionalizing Six Sigma required both specific and common measures, which involved the following:-

Six Sigma organization, launch of the Mission Quality department in 1997 as an umbrella covering all approaches to Six Sigma, to facilitate the Six Sigma initiative across the corporation.

Surround methodologies, asset utilization (tools usage, server utilization).

P**roject methodologies**, defect reduction through Transactional Quality Using Six Sigma (TQSS); Define, Measure, Analyze, Improve, Control (DMAIC); Developing Six Sigma Software(DSSS); and Designing for Customer Satisfaction and Manufacturability (DCAM)[29].

Veloci-Q enabled the delivery of quality products and services without slowing down the system i.e. in time and quick delivery. Introduction of Six Sigma concepts increased the customer and business focus Systems of project execution processes. Veloci-Q complies with the Capability Maturity Model Integration for Engineering, Software Engineering, and Integrated Product & Process Development and ISO 9001:2000 frameworks [37].

Figure 2.8: Veloci-Q Integrated Quality System [37]

The Benefits of using veloci-Q

1. On-Time Delivery

Shifting from the "fixed approach" to "customer orientation" through the introduction of Six Sigma concepts, project-shared vision, team charter, and stressing the importance of measurement and statistics has helped Wipro improve the quality and timeliness of project deliverables and, in turn, customer satisfaction.

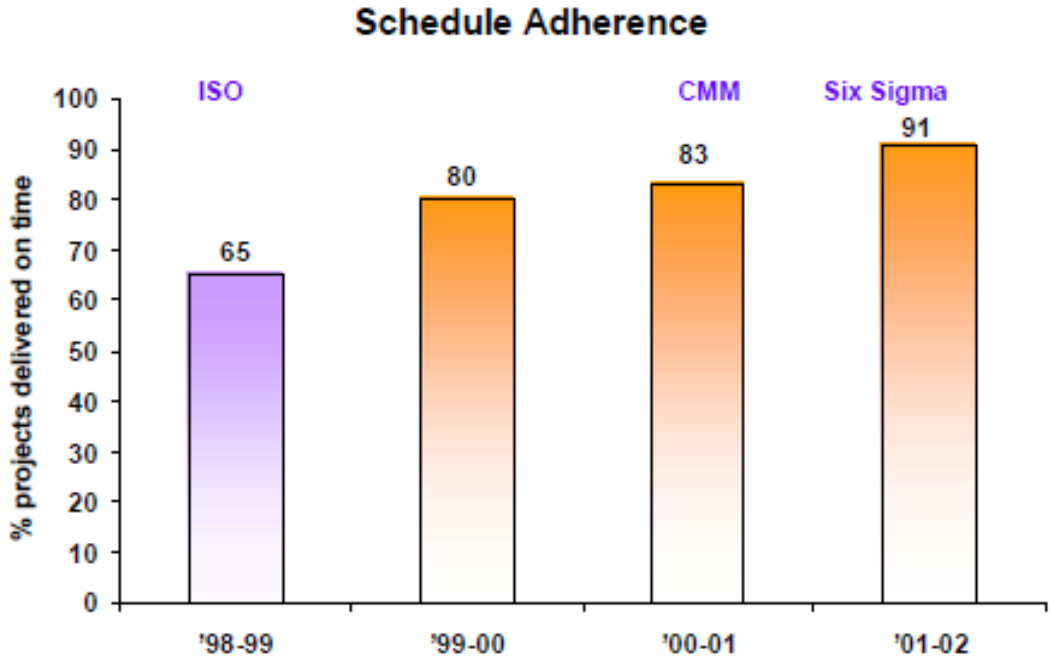

Figure 2.9: Schedule Adherence [37]

2. Improved Quality

The motto of "do it right the first time", focusing on thorough and efficient verification and validation, has resulted in a significant improvement in the quality of deliverables. As shown in Figure 2.10, the introduction of six sigma has resulted in a marked decrease in field error rate (FER). Further, the introduction of new tools in veloci-Q helped in ensuring customer involvement throughout the SDLC, resulting in a significant improvement in product quality. One of the areas consciously focused during the quality journey was phase containment of defects. Many process improvement initiatives that targeted phase containment of defects resulted in reduction of rework and schedule over runs over the period. With the introduction of Six Sigma methodology in software, several Six Sigma projects were undertaken to improve reviews and project management processes during the years 2000-2001.The results of these projects were incorporated in the quality system. The institutionalization of such

practices has led to marked improvement in phase containment of defects between the years 2000-2001 and 2001-2002.

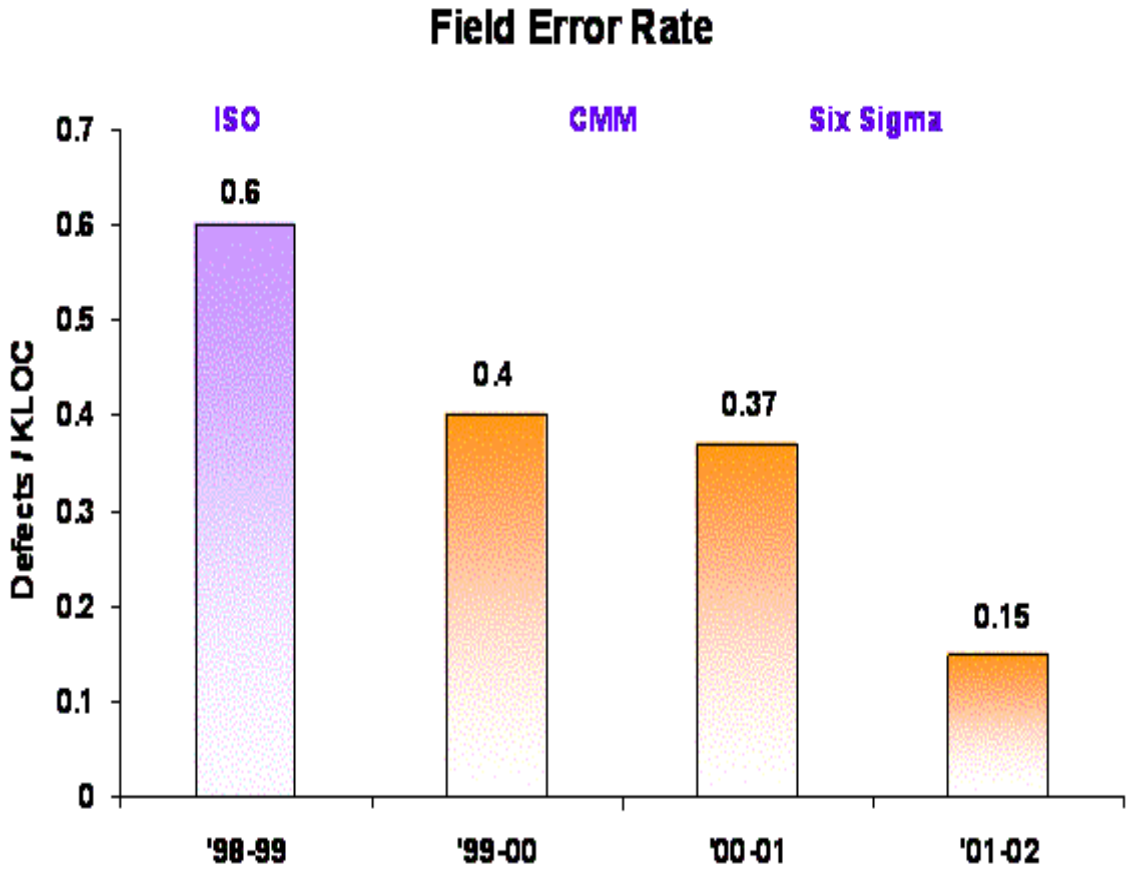

Figure 2.10: Field Error Rate [37]

Computing improvement results or return on investment (ROI) for a process improvement initiative is indeed a complex exercise because (1) the improvement observed (either in productivity or quality of the deliverables) is a result of several parameters, and (2) the investment made on quality has a major influence in enabling the improvement.

Analyzing the results of Wipro Technologies data over the last three years, we have observed the following overall trends:-

i. Productivity has improved by 40% as computed in lines of code/person-day or function points/person-week.
ii. Defect rate on delivered products to customers has been reduced by 60%.
iii. Schedule adherence has improved from 82% to 93% of projects being on time.

The focus on customer-centricity and a data-driven approach to process improvement has played a crucial role in helping Wipro Technologies to achieve its business vision and organizational goals [40].

2.2.2 Process Improvement in Multimodel Environments (PrIME)

The SEI approach is to harmonize the process improvement models. To develop the methods for an organization to harmonize its models—those it uses today and those it might add in the future—the SEI is proposing a three year project called Process Improvement in MultimodelEnvironments (PrIME) [26].

PrIME will think of topics that are needed for an organization to be successful with process improvement in multimodel environments. The project will concentrate on several subsets of models and standards that are commonly used in industry—for example, Six Sigma, CMMI, TQM and many more .The SEI has long held a leading role in process improvement technology research and implementation. This project is an extension of that leadership and the SEI position as an objective, third-party advisor to government and industry organizations.

2.2.3 Process Improvement for Small to Medium Software Enterprises (PRISMS)

PRISMS enables individual SMEs to tailor the software process improvement to the organization's business objectives. It works towards identifying key process areas and customizing and assessing these processes on ordinal scale according to their degree of conformance to the defined process.

The key features of PRISMS process are:-

i. The existing process is examined and if resources permit an explicit model is created.
ii. Early in the PRISMS programme the business goals are defined by management. These goals drive much of the subsequent activity, especially the selection and prioritization of key process areas for improvement, and the selection of measurements.
iii. A consultation exercise is carried out, involving all members of development teams. This is a useful exercise which plays to the strengths of small, flexible teams found in smaller organizations. A brainstorming session, and/or questionnaire-based survey helps the developer team to take ownership of the SPI programme, and to be involved in the programme from the earliest stage.

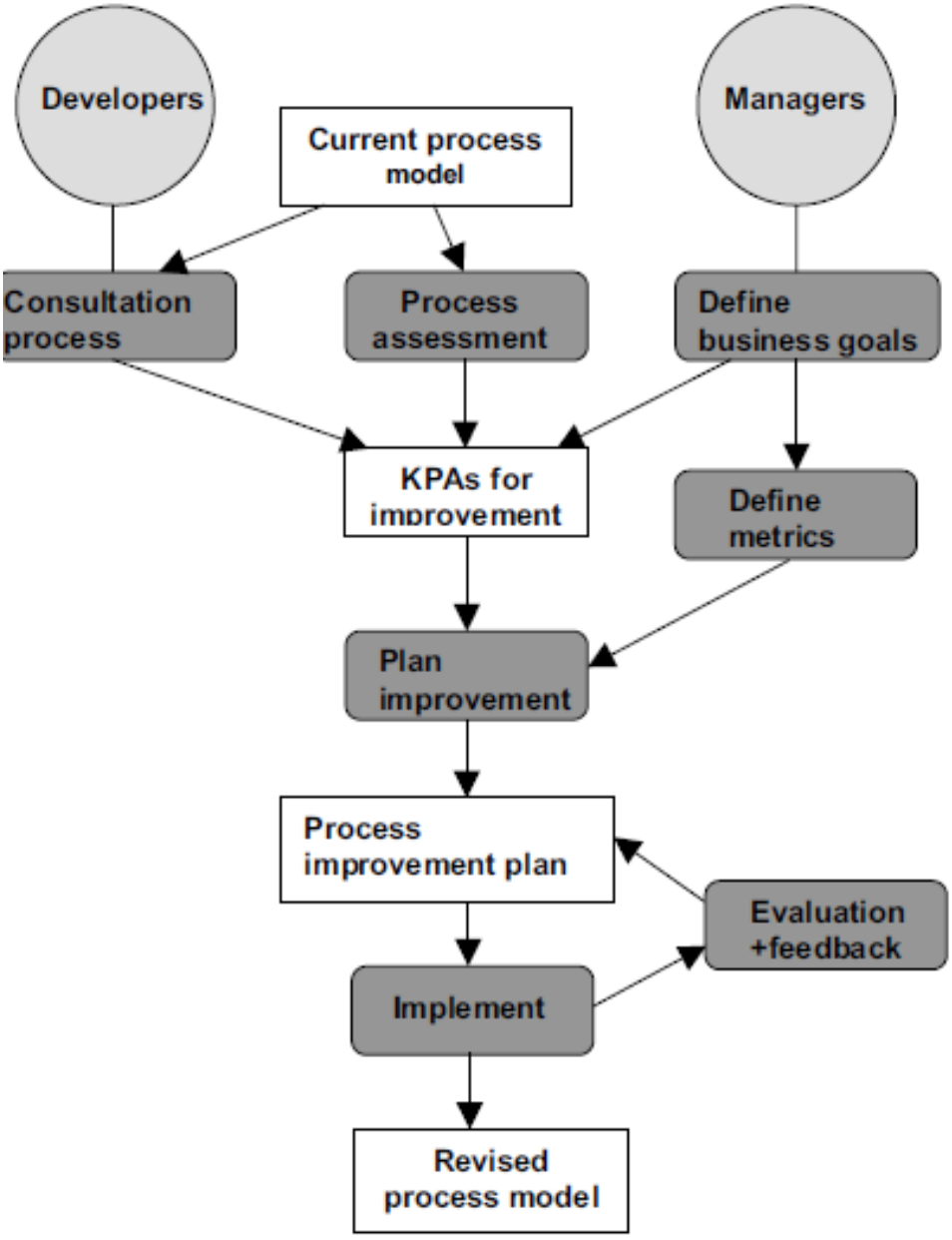

Figure 2.11: The PRISMS Process [5]

iv. A tailored version of the CMM assessment is carried out by members of the research team, primarily to help identify key process areas (KPAs) for improvement. This also indicates the CMM level of the software process, which is often of less immediate usefulness to SMEs, but still useful as a baseline from which to measure future progress.

v. Using these inputs the KPAs for improvement are identified and prioritized. The main criteria here should be the extent to which the KPAs are likely to contribute to the identified business goals.

vi. Measurements are defined as an integral part of the SPI planning process.

vii. The SPI plan is periodically reviewed, and mechanisms are put in place to collect feedback from stakeholders.

a. Assessment Method

Approach to process assessment consists of an *awareness and business case workshop* which focuses on process improvement, cost-effectiveness, and providing a road-map for process improvement, followed by a series of assessment interviews where we meet different groups individually and collectively. Assessment is based on a modified and customizable version of the CMM assessment questionnaire.

b. Measurement

Measurement is an essential part of the PRISMS approach, and managers are keen to have more precise ways of tracking key resource and quality indicators, for example
Goal Question Metric paradigm can be used for the selection of attributes to measure based on the business goals defined for the SPI program [5].

2.2.4 Software Product Line Practices

A software product line is a set of software- intensive systems that share a common, managed feature set satisfying a particular market segment's specific needs or mission and that are developed from a common set of core assets. Core assets form the basis for the software product line. Core assets often include the architecture, reusable software components, requirements statements, documentation and specifications, performance models, schedules, budgets, test plans, test cases, work plans, and process descriptions. The architecture is key among the collection of core assets. Each system in the product line is a product in its own right. However, it is created by taking applicable components from a common asset base, tailoring them through pre-planned variation mechanisms, adding new components as necessary, and assembling the collection. Every software product line has a predefined guide or plan that specifies the exact product-building approach [8]. Development is a term used to describe how core assets (and products) can be used for implementation. Software enters an

organization in one of three ways: the organization builds it (from scratch or by mining legacy software), purchases it (largely unchanged, off the shelf), or commissions it (contracts with someone else to develop it especially for them). So, the term development might actually involve building, acquiring, purchasing earlier work, or any combination of these options. Reuse, as a software strategy for decreasing development costs and improving quality, is not a new idea. However, past reuse agendas, which focused on reusing relatively small pieces of code or opportunistically cloning code designed for one system for use in another, have not been profitable. In a software product line approach, reuse is planned, enabled, and enforced. The reusable asset base includes artefacts in software development that are costly to develop from scratch.

a. Essential Activities in Product Line Practices

There are three essential and iterative activities that blend technology and business practices. A product line involves core asset development and product development using the core assets under the technical and organizational *management*. Figure 2.12 illustrates this triad of essential activities. The rotating arrows in Figure 2.12 indicate not only that companies use core assets to develop products but also that revisions of existing core assets or even new core assets might (and most often do) evolve out of product development. The core assets might be developed or procured for later use in product production. There is a strong feedback loop between the core assets and the products [8]. Core assets are refreshed as organizations develop new products. They then track asset use, and the results are fed back to the asset development activity. Technical and organizational managers manage this process carefully at all levels.

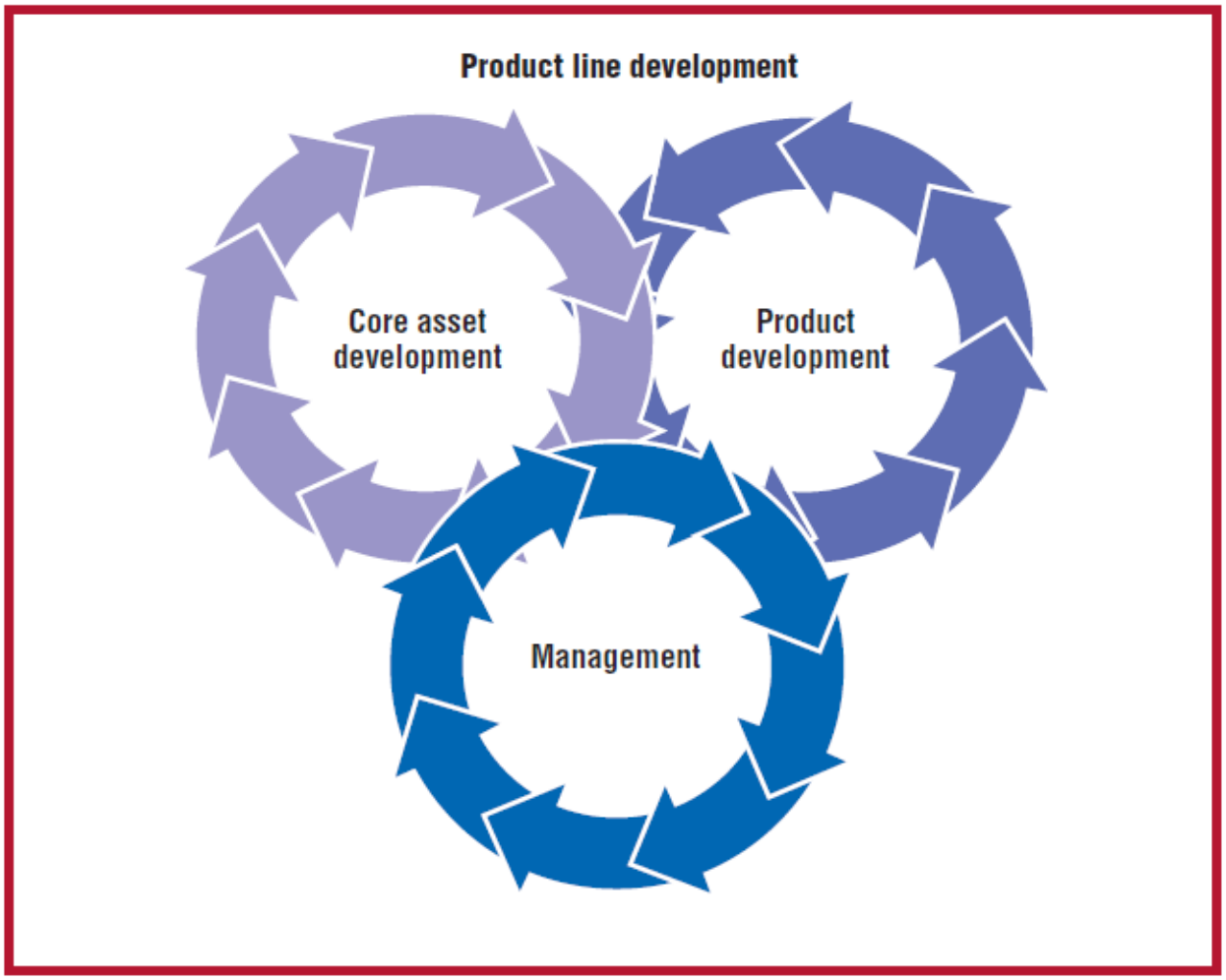

Figure 2.12: Essential product line activities [22]

b. Core Assets Development

The core asset development activity's goal is to establish a production capability for products. Figure 2.13 illustrates this activity, its outputs, and necessary inputs. This activity, like its counterparts, is iterative. Its inputs and outputs affect each other. For example, slightly expanding the product line scope (an output) might admit new classes of systems to examine as possible sources of legacy assets (an input).

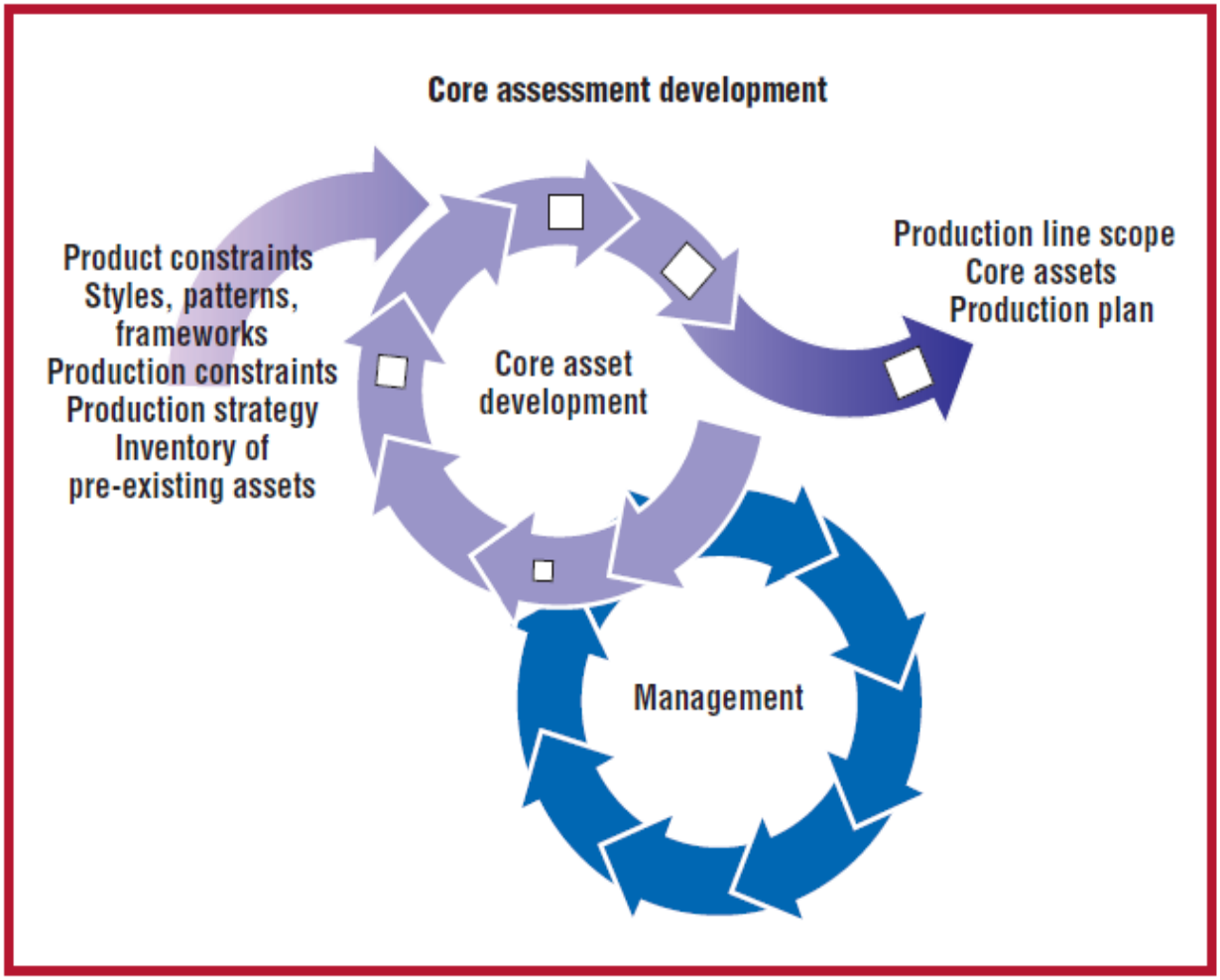

Figure 2.13: Core asset development [22]

Inputs to core asset development include**:-**

Product constraints: Commonalities and variations among the products that will constitute the product line, including their behavioral features.

Styles, patterns, and frameworks: Relevant architectural building blocks that architects can apply during architecture definition toward meeting the product and production constraints.

Production constraints: Commercial or company-specific standards and requirements that apply to the products in the product line.

Production strategy: The overall approach for realizing the core assets. This can be top down (starting with a set of core assets and spinning products off of them), bottom up (starting with a set of products and generalizing their components to produce the product line assets), or combination of both.

Inventory of pre existing assets: Software and organizational assets available at the outset of the product line effort that can be included in the asset base.

c. Product Development

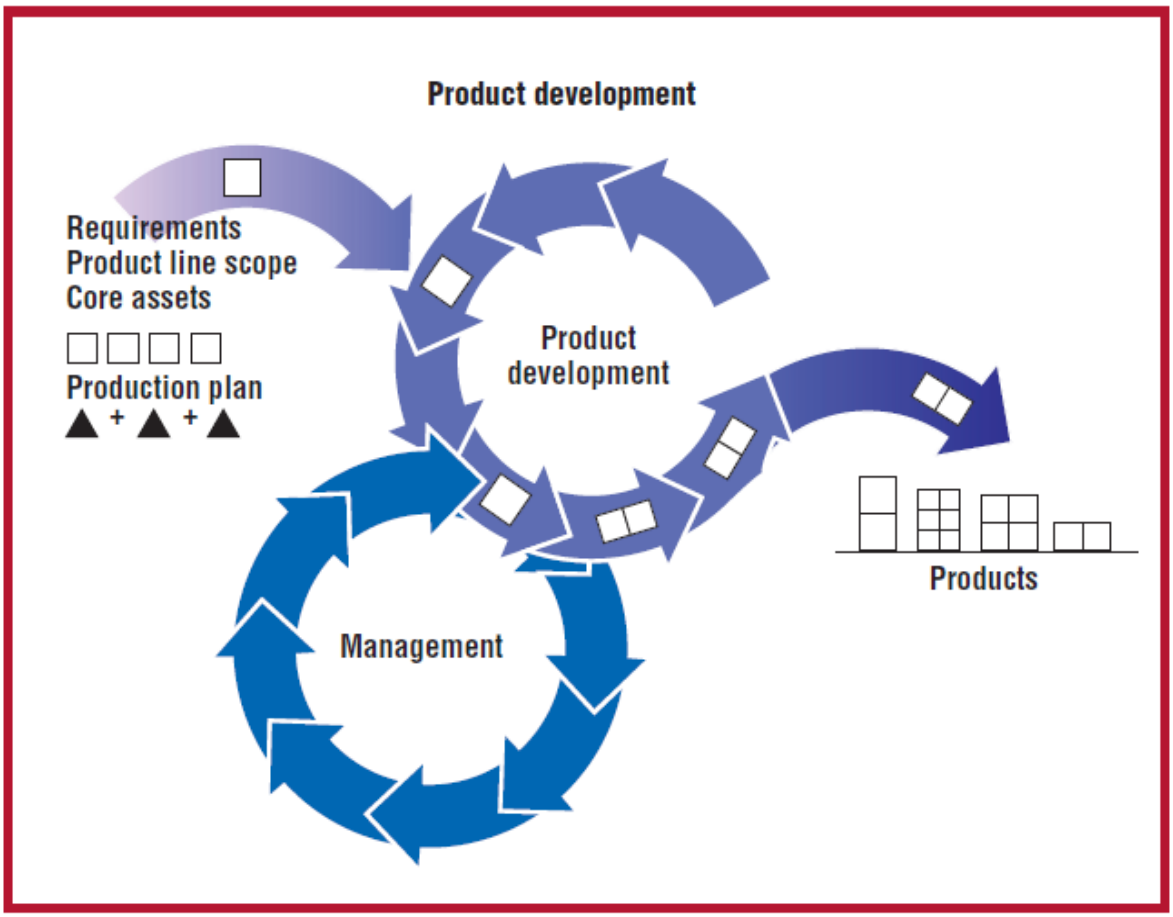

Figure 2.14: Product development [22]

In addition to the three outputs, product development activity depends on the requirements for individual products. Figure 2.14 illustrates these relationships; the rotating arrows indicate iteration. For example, the existence and availability of a particular product might affect a subsequent product's requirements. Creating products can have a strong feedback effect on the core assets, production plan, and even the requirements for specific products. Product development can vary greatly depending on the assets, production plan, and organizational context.

d. Management

Management at the technical (or project) and organizational (or enterprise) levels must be strongly committed to the software product line effort for the product line's success. Technical management oversees the core asset development and the product development

activities, ensuring that the groups building core assets and those building products engage in the required activities, follow the processes defined for the product line, and collect data sufficient to track progress. Organizational management must set in place the proper organizational structure that makes sense for the enterprise and ensure that organizational units receive the right resources (for example, well-trained personnel) in sufficient amounts. Organizational management determines a funding model that ensures core asset evolution and then provides the funds accordingly. It also controls the technical activities and iterations between core asset development and product development. Management should ensure that these operations and the product line effort's communication paths are documented in an operational concept [22]. Management mitigates risks at the organizational level that threaten a product line's success. Product lines tend to engender different relationships with an organization's customers and suppliers, and these new relationships must be introduced, nurtured, and strengthened. Management must create an adoption plan that describes the organization's desired state (that is, routinely producing products in the product lines) and a strategy for achieving that state.

e. Software Product Line Practices

Beneath the surface of the three essential activities are 29 practice areas that our experience shows must be mastered for a successful product line. A *practice area* is a body of work or a collection of activities. They help make the three essential activities more achievable by defining activities that are smaller and more tractable than a broad imperative such as "Develop core assets."In a product line context, each takes on particular significance must be carried out in a unique way. For example, configuration management, an identified practice area, is important for any software development effort. However, configuration management for product lines is more complex than for single systems, those developed one at a time versus using a product line approach. The core assets constitute a configuration that needs to be managed; each product in the product line constitutes a configuration that must be managed, and managing all of these configurations must be coordinated under a single process.

Categorization of each practice area as software engineering, technical management, or organizational management, according to the skills required to carry it out.

f. Software Engineering Practice Areas

Software engineering practice areas are those that are necessary for applying the appropriate technology to create and evolve core assets and products. They are:-

i. Architecture Definition
ii. Architecture Evaluation
iii. Component Development
iv. COTS Utilization
v. Mining Existing Assets
vi. Requirements Engineering
vii. Software System Integration
viii. Testing
ix. Understanding Relevant Domains

Domain understanding feeds requirements, which drive an architecture, which specifies components. Components can be made in-house, bought on the open market, mined from legacy assets, or commissioned under contract. This choice depends on the availability of in house talent and resources, open-market components, an exploitable legacy base, and able contractors. Their existence (or nonexistence) can affect the product line's requirements and architecture. Once available, the components must be integrated and, along with the system, be tested.

Chapter 3

Problem Statement

3.1 Problem Definition

Software process improvement is very difficult to be carried out reason being software development is human based and complex. Because of intangible nature of software, planning & controlling cannot be done as easily as it can be in some other fields of engineering. While doing process changes, we are making changes to current situation which is a very tedious task. Even some of the telecom based companies admit that their success now-a-days is dependent on their strength to implement the software effectively. To implement these continuous changes, transformation is also mandatory in software development models, methods & tools and in management & control of software development as well. Companies want to be more productive and competitive by improving their software process. Software Process Improvement being very advantageous to companies also has challenges & risks associated with its implementation mainly in terms of resources, time and cost.

3.2 Objectives

1. To study Software Process Improvement.
2. To study process areas of software development process and their associated activities.
3. To find out the activities dependent on other activities.
4. Provide a tool for customized software development to improve the process.

Chapter 4

Software Process Customization and Improvement Tool

4.1 Software Process Customization and Improvement (SPCI) Tool

There is always a need to Tailor and Customize the Software Development Process according to the capabilities of the Organization and requirements of the software product to be developed. A Software Development Process can be used in large domain and numerous projects, but we need to optimize the resources to incur minimum cost and maximum utility.

SPCI Tool has been divided mainly in two parts "Existing Model" and "Customized Model" "Existing Models" interface provides Organization a framework where it can select the processes needed to be available in its Software Development Process. As soon as company is clear about the processes needed, it can check the processes using chekboxes and the best suited Software Development Model from available models will be suggested

The Customization has been achieved by providing Organization an interface from which activities needed in the Software Development Process are selected and if the activity selected needs any other prior activity to be executed first then that will be displayed to the user. Organization has been given freedom to select the sequencing of activities needed to be performed but, user needs to have some knowledge of activities and Software Development Process to attain good quality of Software Product using Software Process Customization.

4.1.1 Components of SPCI Tool

The tool can be divided mainly into two main components namely "SPCI Interface" as front end developed using Java & Netbeans and "Database" as backend developed using MySql as backend.

a. **SPCI Interface:** SPCI interface provides user an access to the various modules of SPCI tool. All the modules are explained later on in section 4.2. These Modules/ Forms are:-

i. "Main" Form to give user an option to choose from current available models or to go for a customized model.

ii. "Existing Process Models" form to ask user either to add a new model to Database or to select a model from existing models or to delete an outdated model.

iii. "Add model name to Database" form to add model name to Database by selecting the Process areas to be included in the model from the characteristics options available.

iv. "Delete Model" form to delete any model from the Database along with the Process Areas stored in it.

v. "Selecting model from Process Areas" form suggests the best suited model from the information collected by considering the Process Areas needed.

vi. "Select Activities to be Included" form collects the activities needed in Software Development Process which then navigates to another form which shows the activities needed to be performed before the activities selected by the user and also provides an interface for sequencing the activities.

b. Database: Database has been categorized into four tables. Snapshots of these tables have been depicted later on in table 4.1.Tables are:-

i. "Activities" table showing act_id and act_name fields storing the names of activities along with its unique id.

ii. "Char_activities" table contains the names of Process Areas along with the activities needed to accomplish that Process Area.

iii. "Dependency" table stores the unique id of activities and prior activities needed to be performed if any.

iv. Model_details" table stores the name of model along with the Process Areas associated with it.

4.1.2 Procedure for Customizing a Software Process

When a software development company decides to customize a process, it is important to make a thorough reflection on the concrete situation of the company and study several process models to select one that is suitable. For customizing a model for a company, it is a good idea, to construct a graphical model of the process [6]. This graphical representation will be helpful and useful to find inconsistent, missing or superfluous parts, and to clarify the interactions between activities, roles etc. In addition, the graphical models of the finished software process serves as basis for communicating it during personnel training, for reusing parts in common with other software processes, and for evaluating the software process. Figure 4.1 shows the procedure to make the customizing of a process models with an activity diagram. The activities are described in detail as follows:-

i. **Analyze company's Maturity Level**. The context of a company is analyzed by examining the type of products that are developed, the maturity degree, the personnel capabilities, the current empirical process that it is followed, the products that are generated by the process, and the form in which the projects are managed.

ii. **Study software development models.** Some of the available software process models are the Unified Process, TSPi, PSP, RUP etc. Each model has been created to be adapted and conformed to specific needs.

iii. **Specify development model to be followed**. Modern software processes have taken an iterative and incremental development approach, which reports great benefits. As part of the life-cycle of the software process, it should define phases and their goals. For each iteration, define milestones to evaluate the goals have been reached.

iv. **Define activities.** Each phase has several activities. For each activity is defined in detail: name, goal, steps, input and output artifacts, the role in charge, milestones to evaluate it, type: *(a)* development, *(b)* administration, *(c)* quality assurance, *(d)* communication. If some activity of a software process models is not carried out in the customizing, it is necessary to expose the reasons. Some activities cannot be considered explicitly during the customizing, but its goals must be considered as part of other activity.

v. **Define roles.** *Role* is a set of activities that developers must carry out, so that responsibilities of each person are exactly settled down, according to the role that that person plays at a certain moment. Each role needs abilities from the person who will fulfil it, These responsabilities and a role guide, could be taken from role definition in the software process model. TSPi has defined the administrative role, and the Unified Process defines with greater detail the development role. Once the phases, activities, artifacts, and roles have been defined, the software process is graphically modeled. If the software process is completed, a revision is made for checking if each phase has well defined activities, if parallel activities are allowed, if indispensable roles and desirable ones are identified according to the activities of the software process etc. The revision also covers a review between the input and output artifacts of all the activities. Templates and formats for each artifact are constructed. The tools for each activity are selected: text editors, graphical modelling editors, database management, compilers, supports for Web pages, planning tools for the project, software configuration

management tools. The options for each concept take into account the needs from the company and the types of development. Such options should be clearly settled down.

vi. **Document process.** A graphical model constitutes a documentation of the software process. This documentation serves for guiding the software process, training new personnel, and as a basis for plan improving. It would be convenient to use a tool, that guides the software process in which each graphical element links to the details of the software process.

vii. **Prove in projects.** The software process is tested by using it in a running project, making the necessary adjustments. The selected project does not have to be very simple nor a critical one for the company. The participants in the project need to be voluntary and well motivated. All the result are documented and fixed in the process

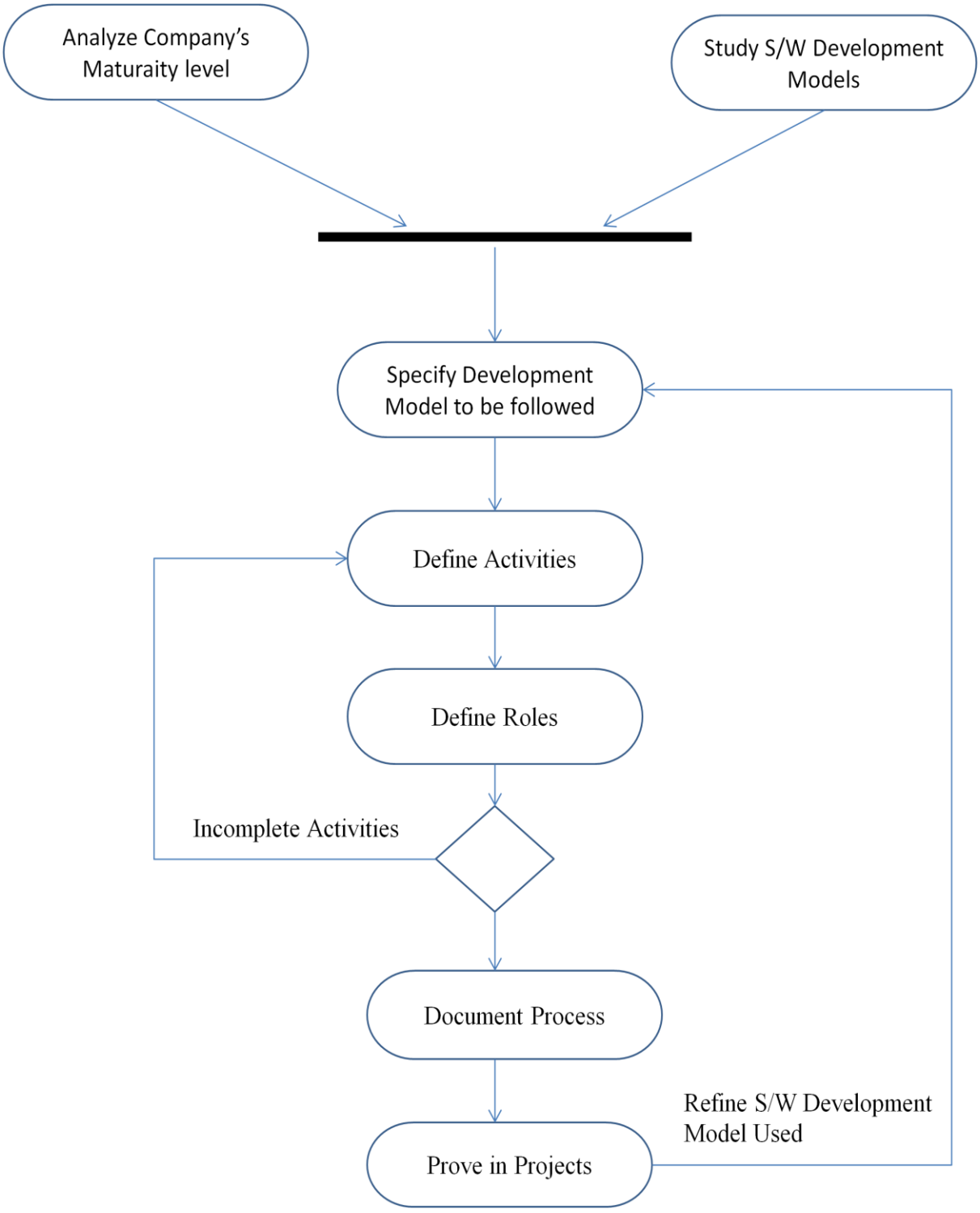

Figure4.1: Procedure of customizing a software process

4.2 Implementation of SPCI tool

The SPCI tool has been developed by finding out the Process Areas being used in a Software Development. A mySql Database of the Process Areas is created with all the activities associated with the Process Areas and mandatory activities to help the Organization/User to develop a quality software product using the best practices possible with optimum use of resources available. Various forms have been shown in the sequence of steps followed in development of SPCI tool:-

4.2.1 Main

Main Form has been designed to provide the interface to the user from where he can start with the working of various functionalities of the SPCI (Software Process Customization and Improvement) tool. There are two buttons provided in this form:-

i. Choose from existing models button: This button connects to a form where functionalities related to the existing models have been shown and provides options to perform the operations on existing Software Development Process Models on the basis of their Process Areas.

ii. Go for customization button: If the user clicks on this button, form will navigate to a form showing the activities already stored in Database and user can select among them the activities needed by him to be used in the Software Development Process being followed to develop the Software Product.

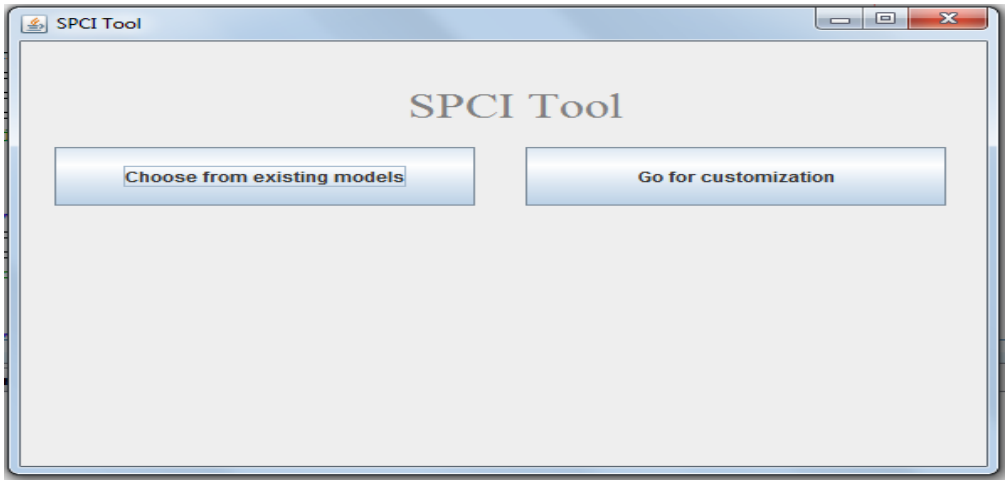

Figure 4.2: SPCI tool interface

4.2.2 Existing Process Models

This form gives the interface to perform following operations:-

i. Add Model: This button navigates user to a form where Process Areas can be selected to be added to a Model, Model name is then stored in the Database along with the Process Areas associated with it.

ii. Select Model: "Select Model" button is used to connect to a form where user can provide the Process Areas needed to be followed in the Software Development Process and SPCI tool will suggest the best possible existing Software Process Model according to the Process Area selected by user.

iii. Delete Model: This button has been included to incorporate the functionality of removing the outdated Models from the Database.

iv. Previous: "Previous" button has the functionality of going back to the form just previous to the current form.

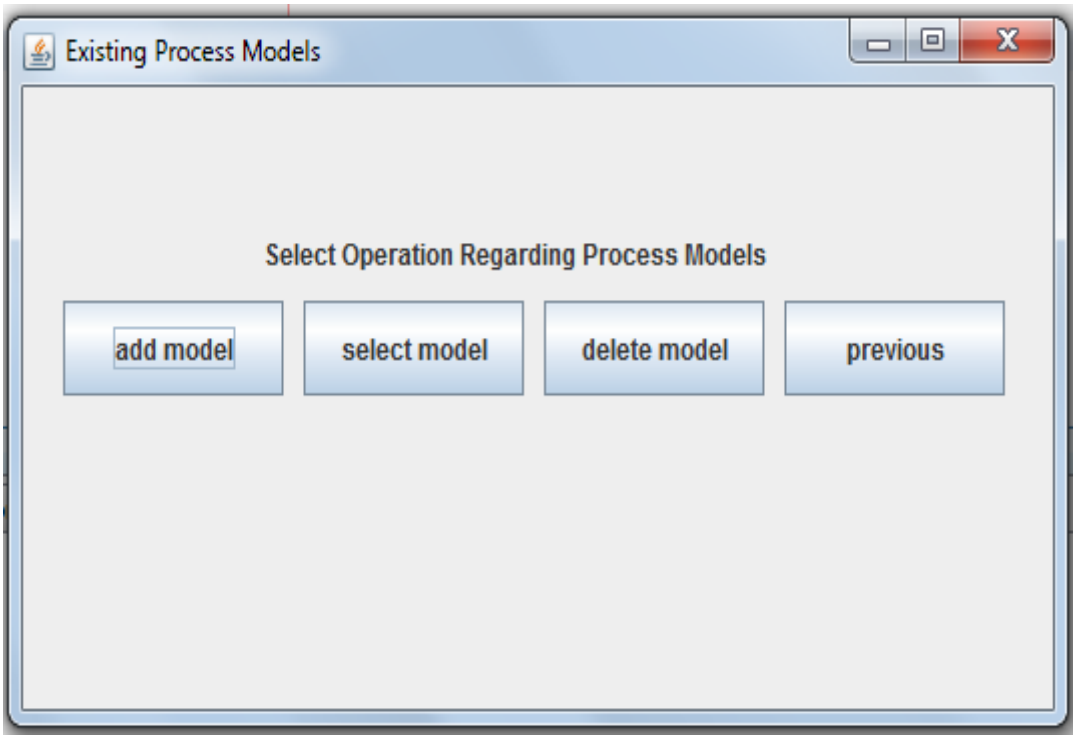

Figure 4.3: Selecting Operations Regarding Process Models

4.2.3 Add Model Name to Database

This form has been designed and developed to add a new model name to the database with the Process Areas to be included in that model as selected by the user. Reset button has been embedded to provide user with functionality to clear all the fields of the form so that user can start afresh when he needs to add another model to the Database. "Previous" button has same functionality as described previously of navigating to the just previous page.

Figure 4.4 shows successful insertion of Model name in Database with corresponding Process Areas as selected by the user. While Figure 4.5 & figure 4.6 ask the user to select the characteristics/Process Areas and to enter Model name respectively.

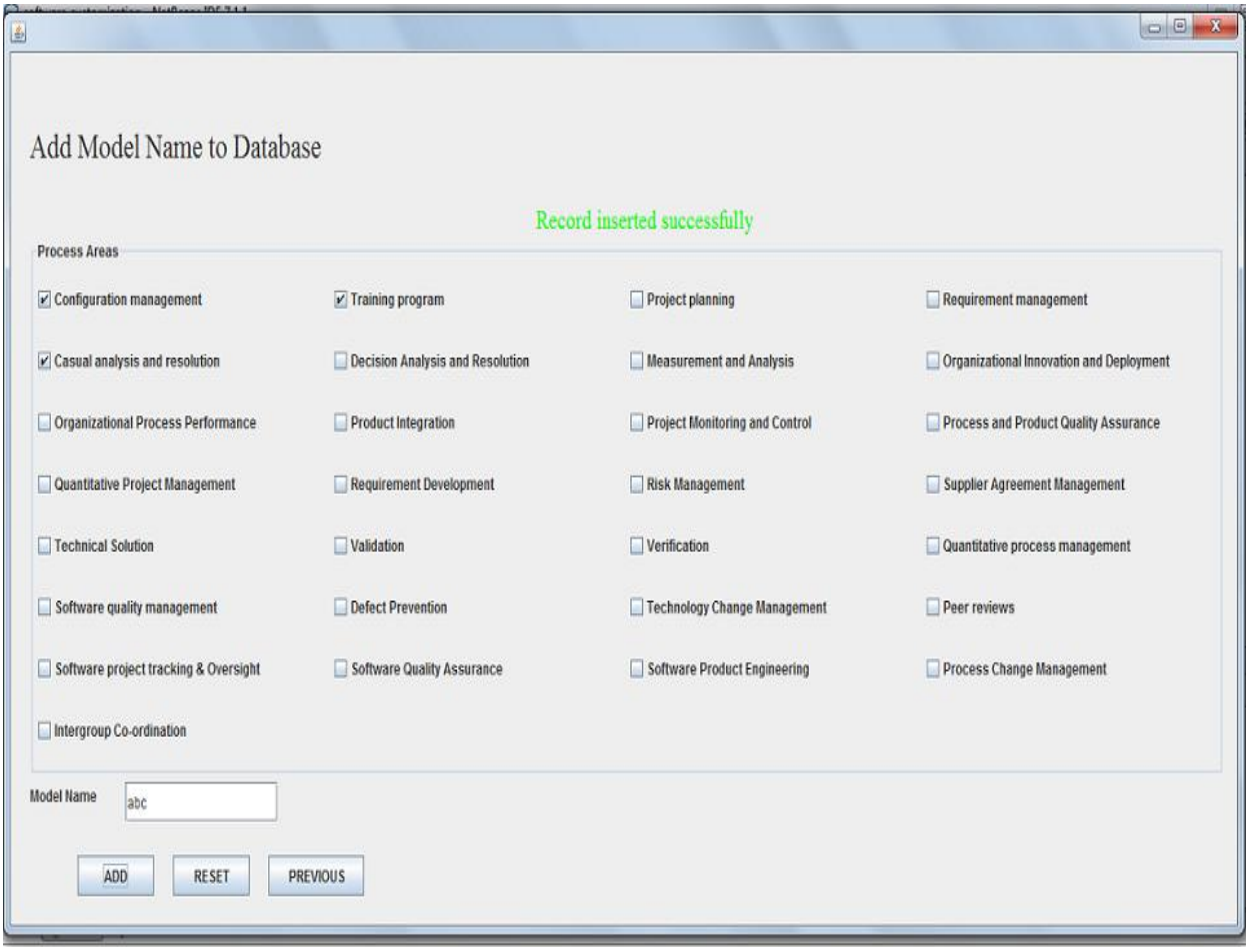

Figure 4.4: Addition of Model name in the Database

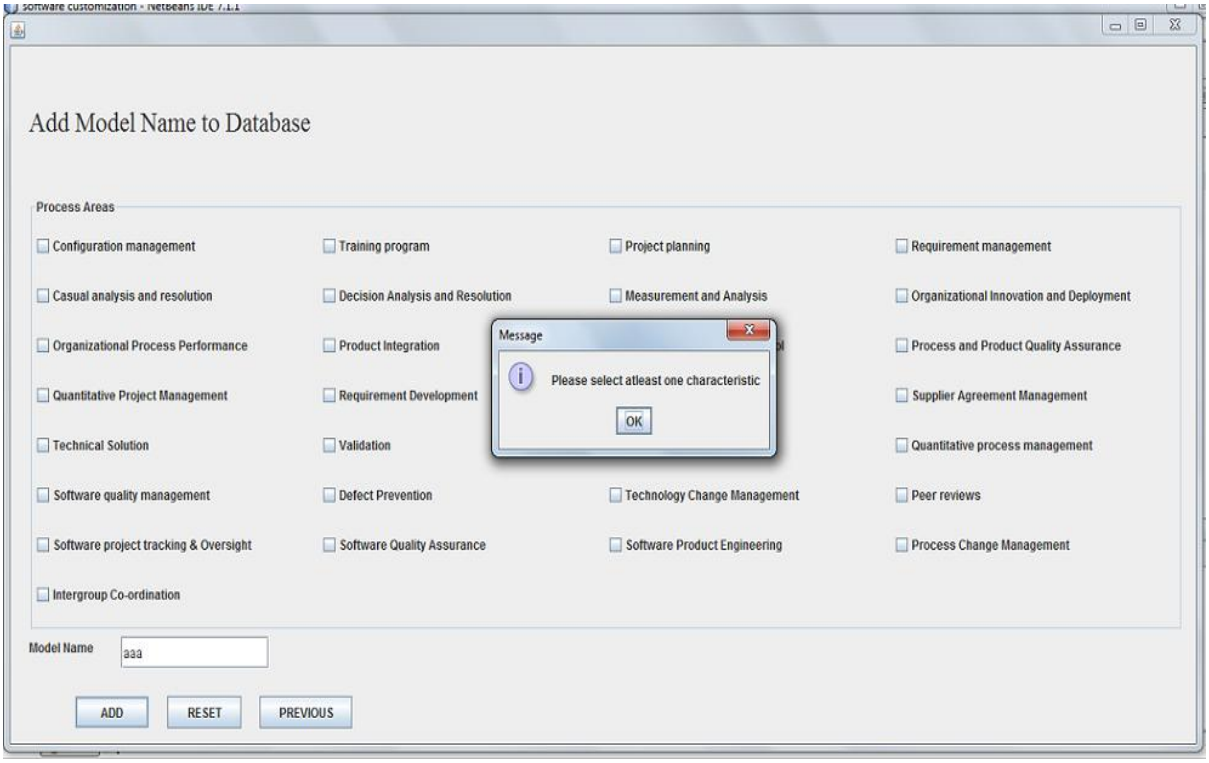

Figure 4.5: Unsuccessful insertion as no Process Area has been selected

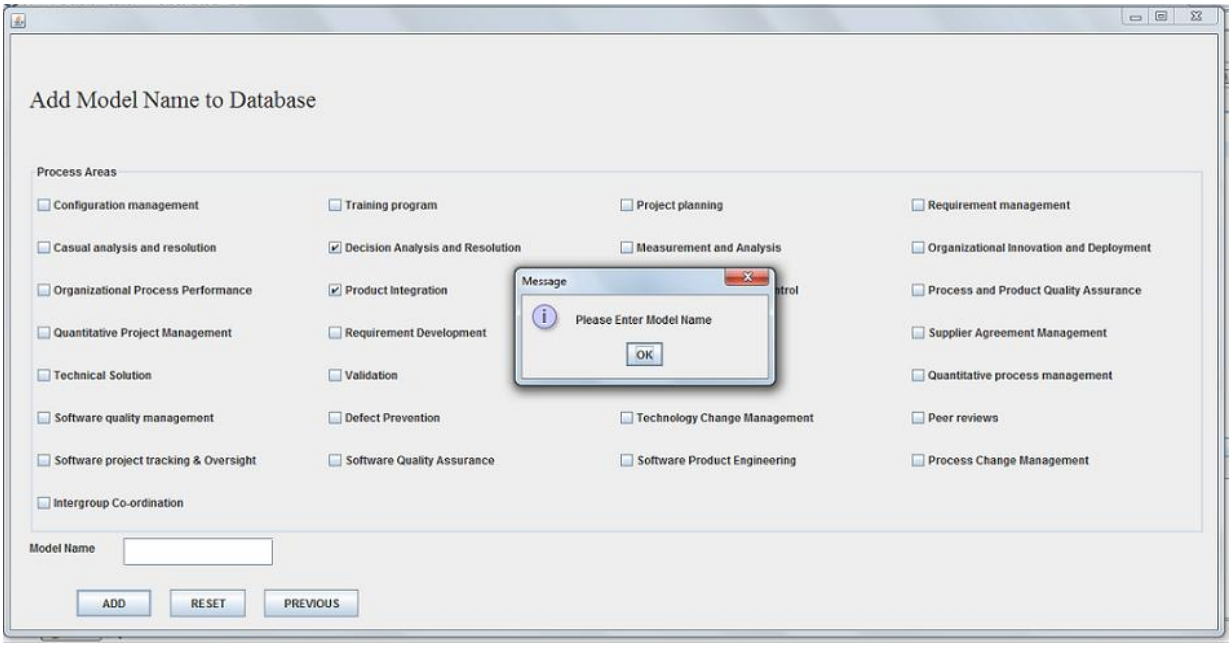

Figure 4.6: Unsuccessful insertion as no Model name has been entered

Figure 4.7 shows how the characteristics of the model have been stored corresponding to a model name and every time we want to add new model, Model_name field will contain the name of the model with associated Process Areas in Model_chars field.

	Model_name	Model_chars
	CMM	Process change management,Technology change manage...
	PSP	Process change management,Technology change manage...
	TSP	Process change management,Technology change manage...
	Six Sigma	Integrated software management,Training program,Or...
	Cmmi	Organizational Innovation and Deployment,Casual an...
	harsh spi	Product Integration,Casual analysis and resolution...
	spi model1	Organizational Process Performance,Casual analysis...
	spi model 2	Decision Analysis and Resolution ,Training program...

Figure 4.7: Database view of model along with its characteristics

4.2.4 Delete Model

Delete form has been developed to provide user with an interface to delete a model from the database. If a model has been deleted then its corresponding Process Areas will also be removed from database. Figure 4.8 & figure 4.9 show the possible outcomes when delete operation is performed. Figure 4.8 shows successful deletion of model from database, but if no such record exists then a message will be popped showing that no such record found as shown in figure 4.9.

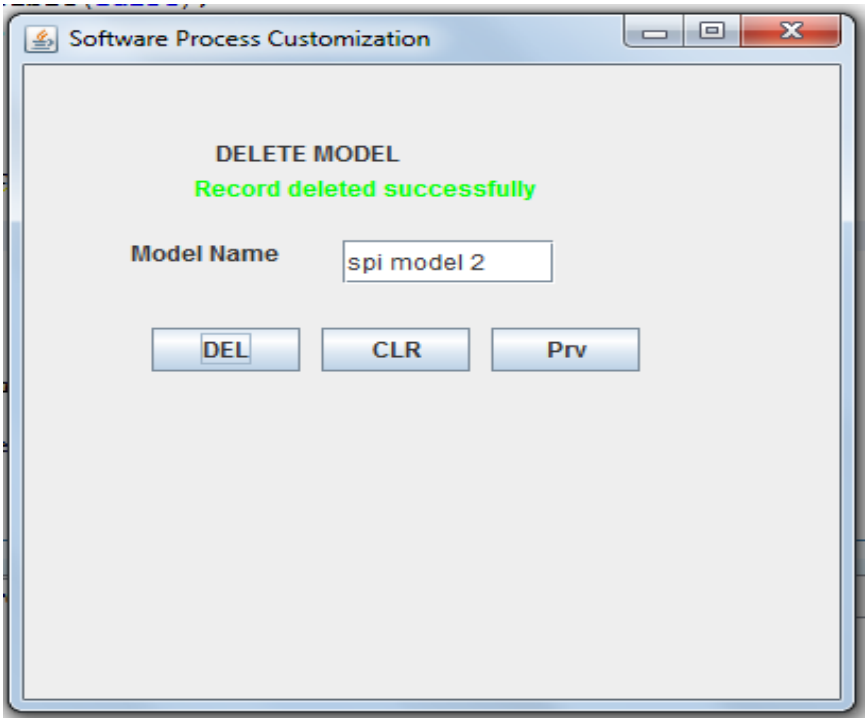

Figure 4.8: Successful deletion of model

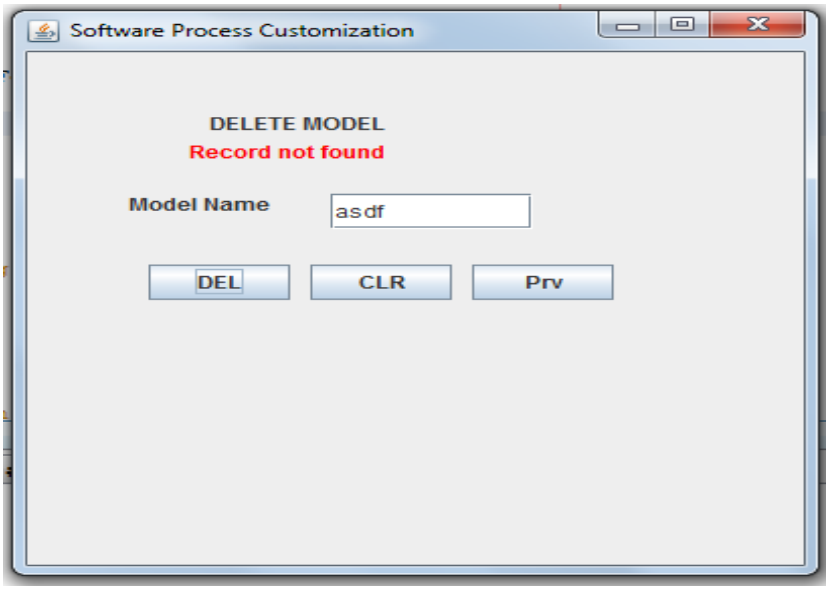

Figure 4.9: Unsuccessful deletion as record not found

S.No.	Process Area	Activity
1	Configuration management	1.1.Identify Configuration items 1.2.Establish a Configuration Management System 1.3.Create or Release Baselines 1.4.Track and Control Changes
2	Training Program	2.1. Determine which training needs are responsibility 2.2. Establish a training plan 2.3. Assess training effectiveness
3	Project Planning	3.1. Estimate the scope of project 3.2. Establish the budget and schedule 3.3. Identify project risks 3.4. Plan for Project resources 3.5. Review plans that affect the project
4	Requirement Management	4.1. Obtain and understand requirements 4.2. Maintain bidirectional tracability of requirements 4.3. Manage requirement changes
5	Casual Analysis and Resolution	5.1. Determine causes of defects 5.2. Implement the action proposal 5.3. Evaluate the effect of changes
6	Decision Analysis and Resolution	6.1. Establish guidelines from decision analysis 6.2. Select evaluation methods for Decision analysis
7	Measurement and Analysis	7.1. Establish Measurement objectives 7.2. Specify measures 7.3. Analyze measurement data 7.4. Store data and result

		7.5. Communicate results
8	Organizational Innovation and Deployment	8.1. Collection and Analyze Improvement Proposals 8.2. Identify and Analyze Innovations 8.3. Manage the Deployment for improvement
9	Organizational Process Performance	9.1. Establish Process Performance Measures 9.2. Establish Process Performance Baselines 9.3. Establish Process Performance Models
10	Product Integration	10.1. Determine Integration Sequence 10.2. Establish Product Integration Procedures and Criteria 10.3. Assemble Product Component
11	Project Monitoring and Control	11.1. Evaluate assembled process Components 11.2. Monitor Project Planning Parameters 11.3. Conduct Progress reviews 11.4. Conduct Milestone Reviews 11.5. Analyze Issues
12	Process and Product Quality Assurance	12.1. Objectively evaluate Processes and Work Products 12.2. Communicate and ensure Resolution of Noncompliance
13	Quantitative Project Management	13.1.Select the sub-processes that will be statistically 13.2. Apply Statistical methods to understand variation 13.3. Monitor Performance of the selected sub-processes
14	Requirement Development	14.1. Develop the Customer Requirements 14.2. Identify Interface Requirements 14.3. Analyze Requirements 14.4. Validate Requirements
15	Risk Management	15.1. Establish a Risk Management Strategy 15.2. Identify Overall risks 15.3. Evaluate, Categorize and Prioritize Risks 15.4. Develop Risk Mitigation Plans 15.5. Implement Risk Mitigation Plans
16	Supplier Agreement Management	16.1. Establish Supplier Agreements 16.2. Execute Supplier Agreements 16.3. Evaluate selected supplier work Products
17	Technical Solution	17.1. Design the Product or Product Component 17.2. Establish a technical data package 17.3.Design Interfaces 17.4. Implement the design
18	Validation	18.1. Select Products for Validation 18.2. Establish Validation procedures and Criteria 18.3. Analyze validation results
19	Verification	19.1. Select Products for validation 19.2. Establish verification procedures and Criteria 19.3. Analyze verification Results

20	Quantitative Process Management	20.1. Document Quantitative Process management 20.2. Follow defined process for quantitative analysis 20.3. Maintain and document process capability baseline
21	Software quality Management	21.1. Develop and document software quality plan 21.2. Maintain Software quality plan 21.3. Define and Monitor Quality goals for product development
22	Defect Prevention	22.1. Plan and Document defect prevention activities 22.2. Identify Causes of defects 22.3.Eliminate causes of defects by prioritizing them
23	Technology Change Management	23.1. Plan incorporation of technology changes 23.2. Evaluate new technologies to determine effects on 23.3 Incorporate feasible new technologies into practice
24	Peer Reviews	24.1. Plan and document peer reviews 24.2. Perform peer review according to document 24.3. Record results of peer reviews
25	Software Project Tracking and Oversight	25.1. Track actual results and performance against 25.2. Take corrective actions if performance deviates 25.3. Take corrective actions if schedule deviates
26	Software Quality Assurance	26.1. Document & plan SQA activities 26.2. Verify adherence of activities to standard 26.3. Inform affected groups about SQA activities
27	Software Product Engineering	27.1. Define and integrate engineering tasks to procedural software 27.2. Maintain consistency of software work products
28	Process Change Management	28.1. Plan continuous process improvement 28.2. Participate in Organization's software process improvement 28.3. Improve software process continuously

Table 4.1: Process Areas and their associated activities [15] [19] [23] [27] [32]

4.2.5 Select Model

This form has been developed to show the activities needed to be accomplished to complete the features of Process Area selected and also to suggest the nearest possible model to fulfill the Process Areas' features. The Process Areas and their associated activities are shown in

Table 4.1 and Figure 4.11. As can be seen in the figure 4.10 & figure 4.12, Submit button's click shows the activities associated with the selected Process Areas and suggested Model name respectively.

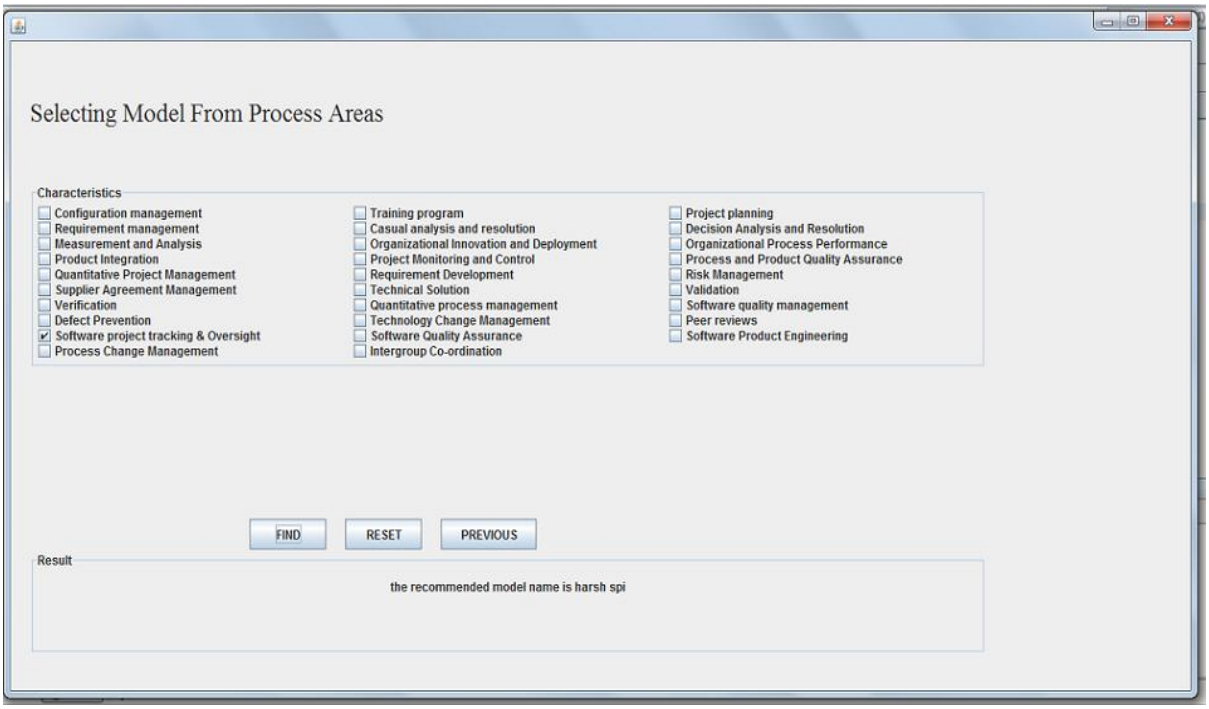

Figure 4.10: Suggested model according to Process Area selected

	Char_name	activities
	Configuration management	Identify configuration items,Establish a configura...
	Training program	Determine which training needs are responsibility ...
	Project planning	Estimate the scope of project,Establish the budget...
	Requirement management	obtain and understand requirements,Maintain bidire...
	Casual analysis and resolution	Determine causes of defects,Implement the action p...
	Decision Analysis and Resolution	Select evaluation methods for Decision analysis,Es...
	Measurement and Analysis	Establish Measurement objectives,Specify measures,...
	Organizational Innovation and Deployment	Collection and Analyze Improvement Proposals,Ident...
	Organizational Process Performance	Establish Process Performance Measures,Establish P...
	Product Integration	Determine Intigration sequence,Establish Product I...
	Project Monitoring and Control	Evaluate Assembled process Components,Monitor Proj...
	Process and Product Quality Assurance	Objectively evaluate Processes and Work Products,C...
	Quantitative Project Management	Select the subprocesses that will be statistically...
	Requirement Development	Develop the Customer Requirements,Identify Interfa...
	Risk Management	Establish a Risk management Strategy,Identify over...
	Supplier Agreement Management	Establish Supplier Agreements,Execute Supplier Agr...
	Technical Solution	Design the Product or Product Component,Establish ...
	Validation	Select Products for Validation,Establish Validatio...
	Verification	Select Products for Verification,Establish Verific...
	Quantitative process management	Document quantitative process management,Follow de...

Figure 4.11: Database view of characteristics and associated activities

Figure 4.11 shows how Process Areas and corresponding Activities have been stored in Database under the field name “char_name” and “activities”.

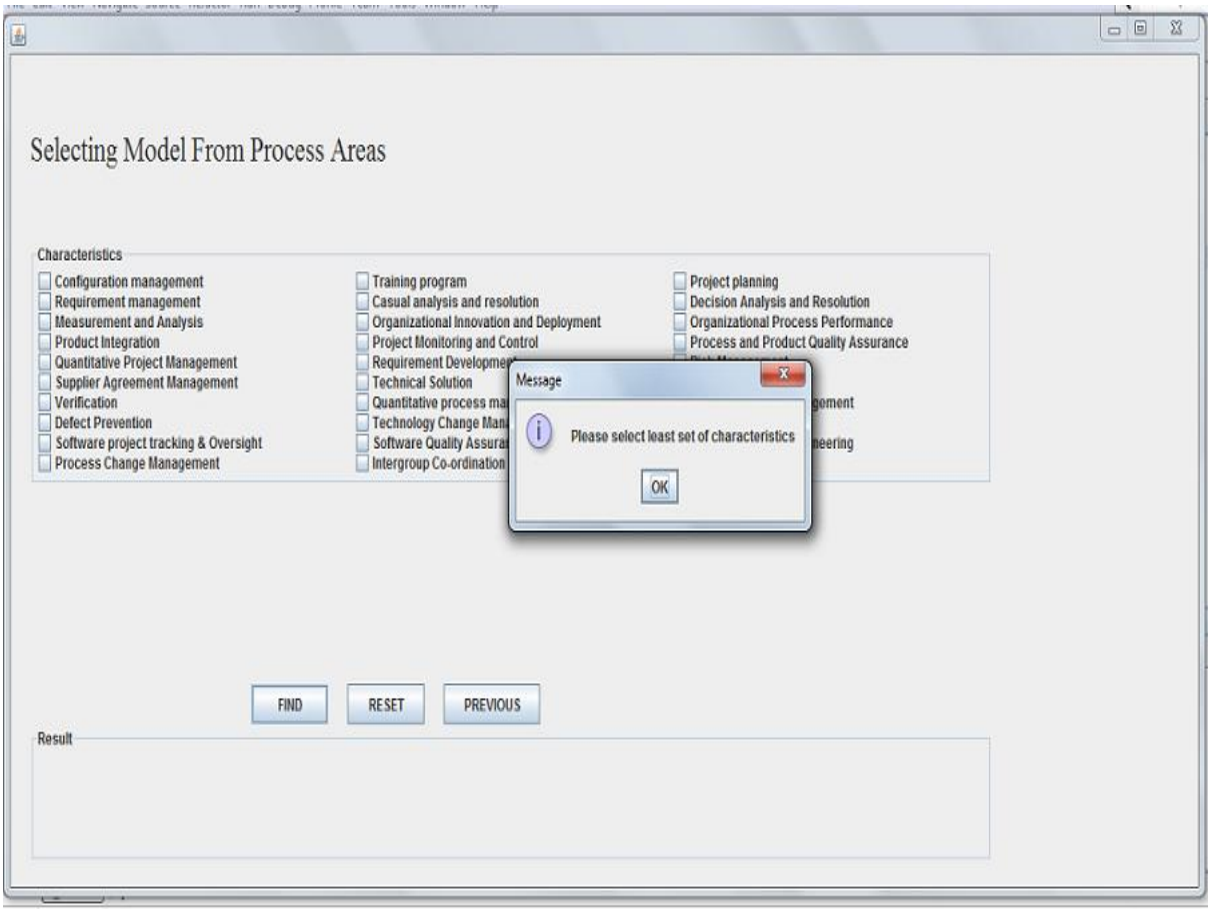

Figure 4.12: Unsuccessful search as none of the Process Areas was selected

4.2.6 Select Activities to be Included

This sub module has been developed to check the activities needed to be completed before accomplishment of the activities selected by the user and also to generate the sequence of activities to be performed according to the priority given by user. To achieve this, all the activities have been assigned a unique id as shown in figure 4.15 where unique id has been given under the name of act_id and then associating act_id with dep_id as shown in figure 4.17 where dep_id means the id of activity which must be performed prior to the activity with corresponding act_id.

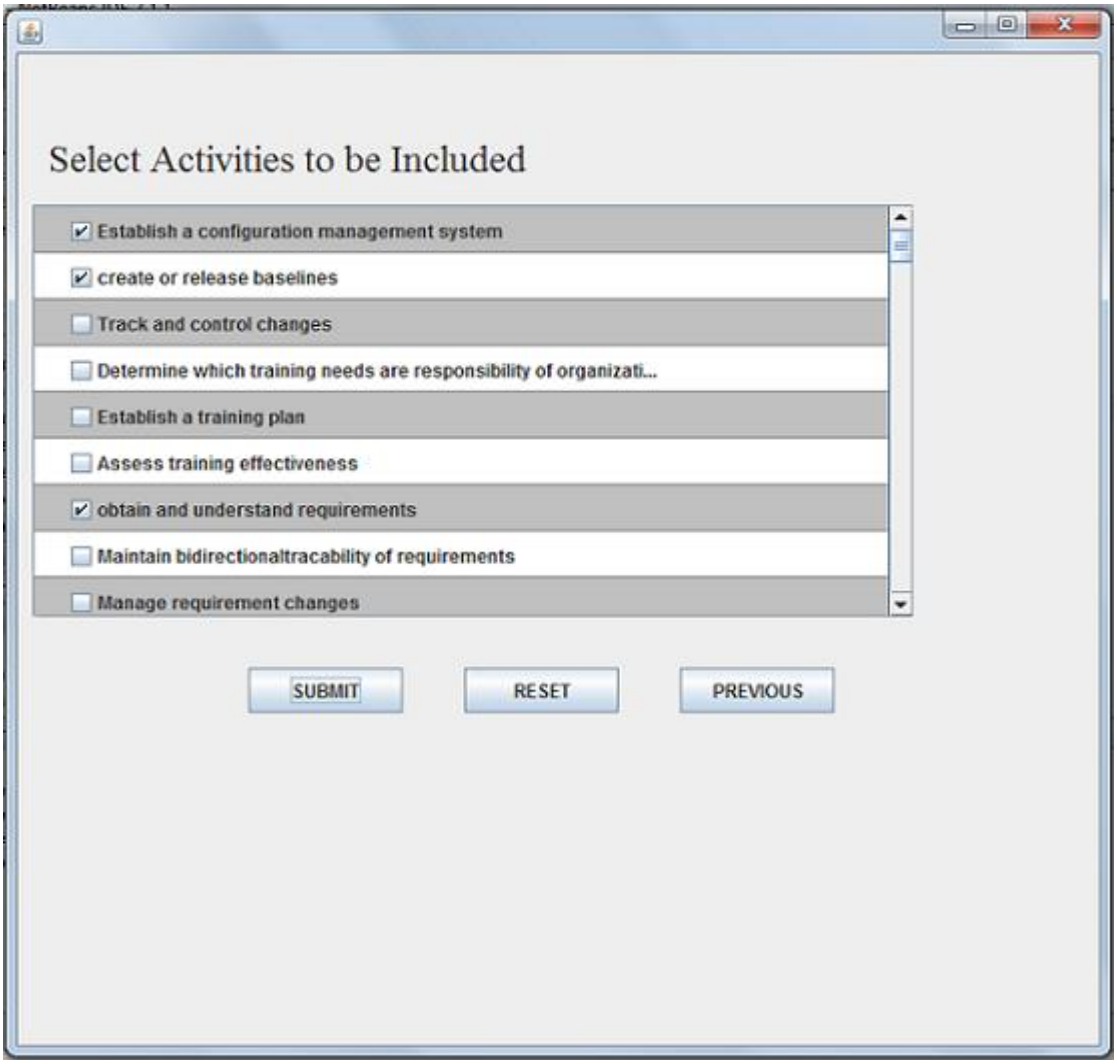

Figure 4.13: Activities to be included in Software Development Process

Figure 4.14 differentiates the activities on the basis of whether the activity is dependent or independent. Dependent activity means we need to perform some other activity prior to the initiation of current activity in the Software Development Process. Here an interface has also been provided to the user where user can select the sequence of activities he wishes to perform in Software Development Process.

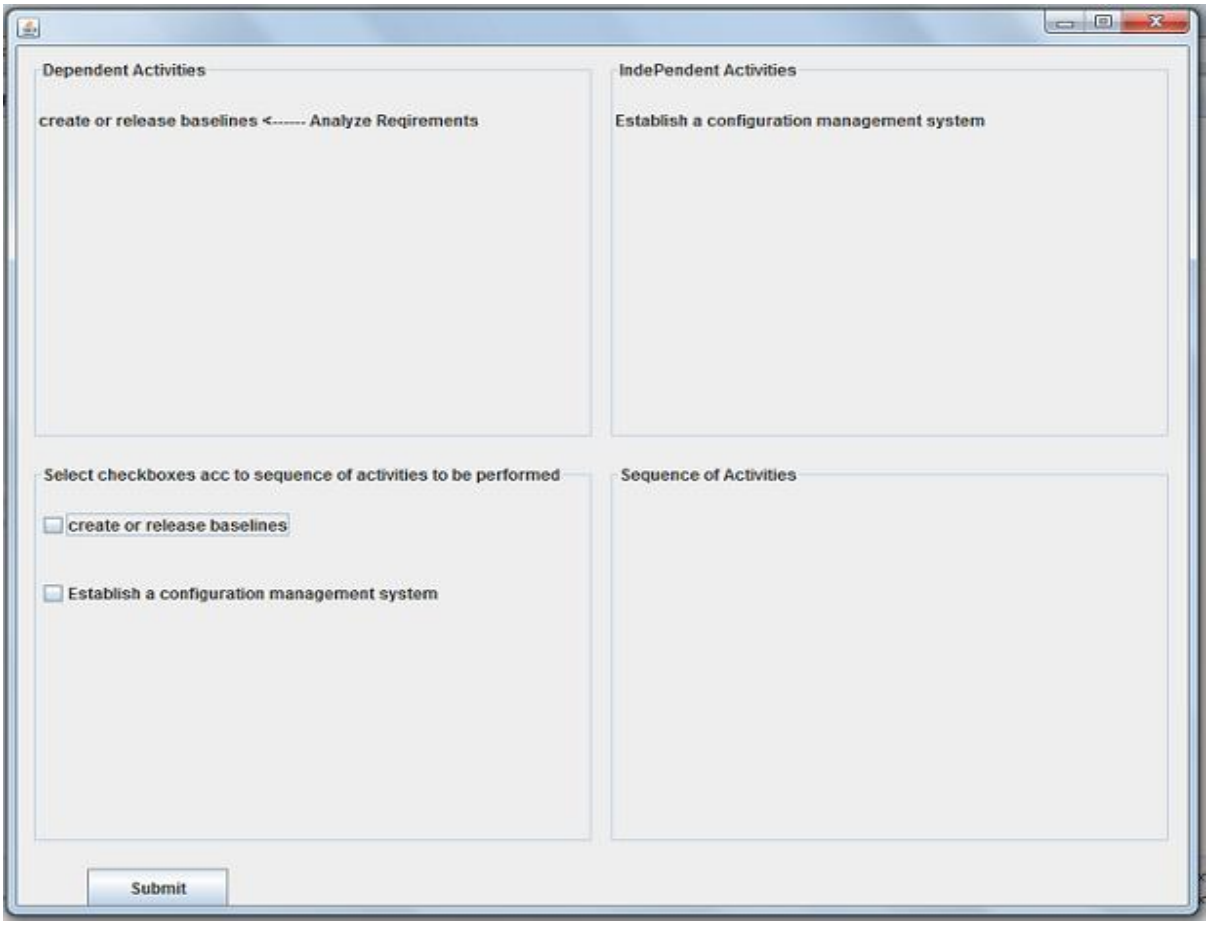

Figure 4.14: Activities selected by user

	act_id	act_name
	SCM1	Identify configuration items
	SCM2	Establish a configuration management system
	SCM3	create or release baselines
	SCM4	Track and control changes
	TP1	Determine which training needs are responsibility ...
	TP2	Establish a training plan
	TP3	Assess training effectiveness
	RM1	obtain and understand requirements
	RM2	Maintain bidirectionaltracability of requirements
	RM3	Manage requirement changes
	PrjP1	Estimate the scope of project
	PrjP2	Establish the budget and schedule
	PrjP3	Identify project risks
	PrjP4	Plan for Project resources
	PrjP5	Review plans that affect the project
	CAR1	Determine causes of defects
	CAR2	Implement the action proposal

Figure 4.15: Database view of activities and their unique id

Figure 4.16 shows the sequence of activities to be performed by the user according to the priority of activities given by him. The sequence is generated with the combination of sequence selected by user while marking check on checkboxes given in "activities" interface of Figure 4.16 and by sequencing the prior activities mandatory to be accomplished in case of dependent activities. Considering the case shown in Figure 4.16 where activities selected by user are "create or release baselines" and "Establish a configuration management system". Both the activities have been differentiated on the ground of Dependent and Independent activities. Third interface asks the user to check the checkboxes in the sequence he wishes activities to be performed in his process. User has given the sequence as follows:-

i.) Establish a configuration management system.

ii.) Create or release baselines.

As can be seen from "Dependent Activities" interface, we need to perform "Analyze Requirements" activity prior to "Create or release baseline. Thus, the sequence of activities displayed in "Sequence of activities" interface is:-

i.) Establish a configuration management system

ii.) Analyze Requirements

iii.) Create or release baselines

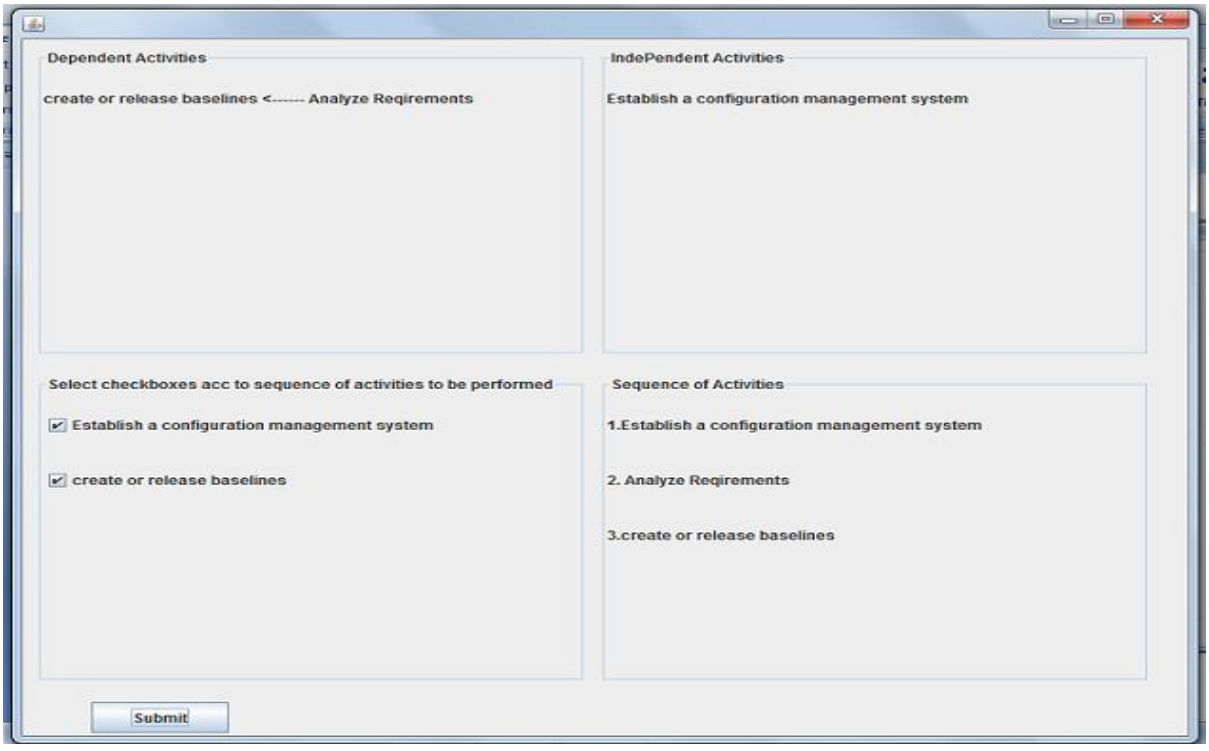

Figure 4.16: Sequencing of activities selected by user

	act_id	dep_id
	SCM1	RM1
	SCM3	RD3
	SCM4	PrjP4
	TP2	PrjP4
	RM1	RD1,RD3
	RM3	SCM4
	PrjP1	RD1
	PrjP2	RD1
	CAR1	RsM2
	CAR3	MA3
	DAR1	RD3,MA3
	OID1	OPP3
	OPP1	MA3
	OPP2	QPrjM1
	PrdI1	RM1
	PrdI2	PrjP5
	PMC1	Val2
	PMC2	PrjP4
	PPQA1	OOP1
	PPQA2	Val3

Figure 4.17: Database view of activity id along with prior activity's id

Chapter 5
Conclusion and Future Scope

5.1 Conclusion

In this thesis work, a customization tool has been developed to provide the user/organization a method to select the process area or activities an organization wants to include in the software development process so that it can have same properties in the software product it is about to develop. Various areas that have been covered are:-

1. To have a best suited already existing model to be followed in software development in accordance with the process areas or characteristics needed in product/project to be developed.
2. Learning has been successfully embedded in customized process by finding the activities dependent on prior occurrence of other activities.
3. Adding more models to the database pool by including process areas related to that model.

5.2 Future Scope

There are many extensions of this work that can be considered in future research. Some of them can be:-

1. Adding numerous process areas to current process areas database will prove to be more helpful to the organizations.
2. Finding more learning techniques will enhance the current tool's capability.
3. Searching activities to a large number will make the tool a prominent method for customizing the software process.

References

[1] Ahn Y.W., Ahn H.J. and Park S.J., "Knowledge and Case-Based reasoning for Customization of software processes*", International Journal of Software Engineering and Knowledge Engineering*, 2003.

[2] Alexendra S., Habra N., Desharnais J.M., Laporte C.Y. and Renault A., "Initiating software process improvement in very small enterprises Experience with a light assessment tool", *ScienceDirect Information and Technology*, 2008

[3] Alexendra S. and Habra N., "UML Modeling of Five Process Maturity Models", *Laboratoire de Qualite logicielle*, version 1, 2003.

[4] Alexendra S., Renault A. and Habra N., "OWPL: A Gradual Approach for Software Process Improvement In SMEs", *ieee* , 2006

[5] Allen P., Ramachandran M. and Abushama H., "PRISMS: an Approach to Software Process Improvement for Small to Medium Enterprises"**,** Leeds Metropolitan University, Beckett Park Campus, 2003.

[6] Bandinelli S., Fuggetta A. and Ghezzi C., "Software Process Model Evolution in the SPADE Environment", GOODSTEP, Rep. 014, 1993.

[7] Bohem B.W., "A spiral model for software development & Enhancement", *IEEE* ,1988

[8] Clements P. and Northrop L., *Software Product Lines: Practices and Patterns*, Addison-Wesley, Boston, 2001.

[9] Condari R. and Fuggetta A., "Improving Software Process Improvement", *IEEE*, vol. 19, Issue 4, 1998

[10] Conradi A., Liu C. and Jaccheri M.L., Process Modeling Paradigms: An Evaluation, Norwegian Institute of Technology (NTH), Trondheim, Norway. Position paper at *7th International Software Process Workshop (ISPW7)*, USA, 1991.

[11] Francisco J.P., Felix G.and Mario P., "Software process improvement in small and medium software software enterprises",*Springer Science*,2007

[12] Harjumaa L., Tervonen I. and Vuorio, "Using Software Inspection as a Catalyst for SPI in a Small Company", *Springer Science*, 2004

[13] Humphrey W.S, "Introduction to Software Process Improvement", *Software Engineering Institute*, Jan 1993.

[14] Ibarguengoitia G., Salazar J.A,Sanchez M.G. and Ramirez A.Y., "A Procedure for Customizing a Software Process", *Conf. Fourth Mexican International Conference*, pp 68-72, 2003.

[15] Jansson A.S, "Software Maintenance and Process Improvement by CMMI", Uppsala University, Sweden, Dec 2007

[16] Jiang J.J, Klein G., Hwang H.G., Hwang J. and Hung S.H, "An exploration of the relationship between software development process maturity and project performance", *ScienceDirect Information & Management*, 2004, pp 279-288.

[17] Krasner H., "The Payoff for Software Process Improvement: What it is and How to Get it", *IEEE Computer Society*, 1999.

[18] Munassar N.M.A and Govardhan A., "A comparison between five models of software engineering", *IJCSI International Journal of Computer Science*, Vol. 7, Issue 5, September 2010.

[19] Mellon C.," Process Improvement in Multimodel Environment", *Software Engineering Institute*,2008.

[20] Muhonen M, "Software Process Improvement: Continuous Integration and Testing for Web Application Development", M.Sc. Thesis, Department of Computer Science , University of Tempere, Finland, June 2009.

[21] Niazi M., Wilison D and Zowghi D, "A maturity model for the implementation of software process improvement", *The Journal of system and software*, 2003.

[22] Northrop L.M, "SEI's software product line tanets", *IEEE*, vol. 19, Issue 4, 2002

[23] Paulk M.C, Curtis B., Chrissis M.B and Weber C.V, "Capability Maturity Model for Software", version 1.1, *Software Engineering Institute*, 1993.

[24] Pettersson F., Ivarsson M., Gorschek T.and Ohman P., "A practitioner's guide to light weight software process assessment and improvement planning", The Journal of System & Software, August 2007.

[25] Pollice G., "Matching Project and Processes", *The Rational Edge*, May 2004.

[26] Process Improvement in Multimodel Environments (PrIME), [online], Available: url [Accessed: October 30, 2011].

[27] Rainer A. and Hall T., "Key success factors for implementing software process improvement: a maturity-based analysis.", *Journal of System and Software*, 2002.

[28] Salo O., "Enabling Software Process Improvement in Agile Software Development Teams and Organizations", *VTT Publications*, 2006

[29] Seckin H., "Software Process Improvement Based On Static Process Evaluation", M. Sc. Thesis, Electrical and Electronics Engineering Deptt., The Graduate School Natural & Applied Sciences of Middle East Technical University, Turkey, April 2006.

[30] Seller B.H., Serour M., McBride T., Perez C.G, and Dagher L., "Process Construction and Customization", *Journel of Universal Computer Science*, vol. 10, 2004

[31] Sirvio S.K., "Development and Evaluation of Software Process Improvement Methods", Faculty of Science, University of Oulu, Finland, 2004.

[32] Software Engineering Institute CMMI process areas, [online], Available: http://www.tutorialspoint.com/cmmi/cmmi-process-areas.htm [Accessed : April 16,2012].

[33] Software Process Improvement, [online], Available: http://www.soberit.hut.fi/T-76.5631/ [Accessed: May 5, 2012].

[34] Sommerville Ian, "*Software Engineering*", 7th ed. ,Addison Wesley, 2004.

[35] Stelzer D., Mellis W. and Herzwurm G., "A critical look at ISO 9000 for software quality management", *Software Quality Journal*, 1997.

[36] Stelzer D. and Mellis W., "Success Factors of Organizational Change in Software Process Improvement" *John Wiley & Sons Ltd*., 1999.

[37] Subramanyam V., Sambuddha Deb, Krishnaswamy P. and Ghosh R., "An Integrated Approach to Software Process Improvement at Wipro Technologies: veloci-Q", *DTIC* 2004

[38] Taya S. And Gupta S., "Comparative Analysis of Software Development Life Cycle Models", IJCST, vol. 2, Issue. 4, Oct-Dec 2011.

[39] The PDCA Improvement process, [online], Available: http://www.logmgt.nkmu.edu.tw [Accessed: May 5, 2012].

[40] VanSolingen R., “Measuring the ROI of Software Process Improvement”, *IEEE*, vol. 21, Issue 3, 2004.

[41] Villalon J.A., Agustin G.C., Gilabert T.S.F, “Experiences in the Application of Software Process Improvement in SMES”, *Software Quality Journal*, 2002.

List of Publications

1. Harsh Taneja, Ashima Singh, 2012, "ISSUES AND CHALLENGES IN INTEGRATION OF AGILE PROCESSES TO TRADITIONAL SOFTWARE PROCESS MODELS", in Proceedings International Conference on Competitiveness & Innovativeness in Engineering, Management & Information Technology (ICCIEMI-2012), January, 2012.

2. Harsh Taneja, Ashima Singh, 2012, "ISSUES AND CHALLENGES IN INTEGRATION OF AGILE PROCESSES TO TRADITIONAL SOFTWARE PROCESS MODELS", International Journal of Research in Engineering and Applied Sciences, ISSN: 2249-3905, Volume 2, Issue 2, pp. 1103-1109, February 2012.