ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

Software Improvement Model for small scale IT Industry

T.B.Patil[1], S.D.Joshi[2], R.M.Jalnekar[3], Manjusha Joshi[4]

M.Tech Student, Department of Computer Science, Bharati Vidyapeeth Deemed University,

College of Engineering, Pune, India[1]

Professor, Bharati Vidyapeeth Deemed University, College of Engineering, Pune, India[2]

Director Vishwakarma Institute of Technology,Pune[3]

Research Scholar, BVDUCOE Pune[4]

Abstract: Over the years, the software development has evolved from just being science to a combination of "art" and "science". Today's software progress environment, follow lifecycles with phases that are either sequential or parallel in execution. The software process is a set of actions, methods and transformations that people use to develop and maintain software and the associated products, for example: product plans, blueprint, code, test cases and user manuals. This paper presents a methodology for assessing software processes which assist the activity of software process improvement in small organizations. There is an effort to address issues such as the fact that: (i) process assessment is expensive and typically requires major company resources and (ii) many light assessment methods do not provide information that is detailed enough for diagnosing and improving processes.

Keywords: Design Phase, Implementation Phase, KPA, Planning Phase, Security Phase, Six Sigma, Software Metric

I. INTRODUCTION

Software process improvement and measurement is becoming one of the main methods to solve "software crisis". Software Process Measurement (Software Metric) defines the process of software development, collects and analysis data, that is quantization process of continuous improvement, is important basis of making plan, executing process, implementing control.

SIX SIGMA[1]

Six sigma strategies were developed by Motorola in the early 1990s. Six Sigma is based on statistical approach which does the improvement by historical data and by calculation of mathematical formulas. The goal of the six sigma is to detect the defect and reduce the defect. [2] Six sigma means a company tries to make "error-free product 99.9997% of the time a minuscule 3.4 errors per million opportunities". SIX SIGMA has six stages and reduces the defect step by step.

Six Sigma is usually related to the magic number of 3.4 defects per million opportunities.

Limitations of Six Sigma: The limitation of Six Sigma can be given as following.

- Six Sigma is a statistically-based process improvement methodology.
- Often it is very difficult for small companies to take employees away from their regular duties in order to be trained in Six Sigma. If employees are not available to give their services, the company loses money due to a reduction in productivity.
- Six Sigma focuses on prioritizing and solving specific problem which are selected based on the strategies priorities of the company and the problems which are causing the most defects.

In this project we have tried to make a simple yet really efficient tool which can be easily used by the developers without any formal training. This will reduce the extra burden which Six Sigma forced on companies.

We have mainly focused on four important phases of software development life cycle:-

i. Planning Phase
ii. Design Phase
iii. Security Phase
iv. Implementation Phase

For each of the above written phases we have designed a simple GUI for testing. Based on the quality questions, answered by the tester KPA (Answering Criteria Percentage) is calculated using an algorithm

KPA [3]

KPA stands for Answering Criteria Percentage which calculates the average success percentage of each phase.

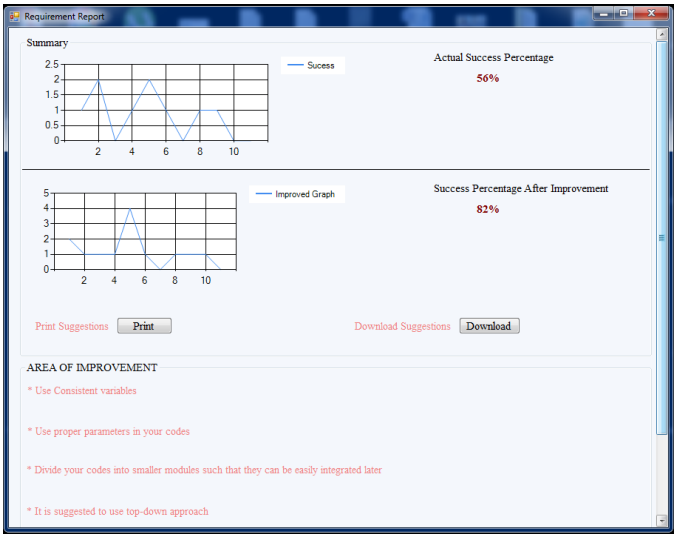
$$KPA = \frac{\text{(No. of answering per criteria)}}{\text{(No. of questions – No. of N/A answer)}}$$

***NOTE**
Number of answer of "*NO*" or "*PARTIALLY*" will be treated as areas for improvement.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

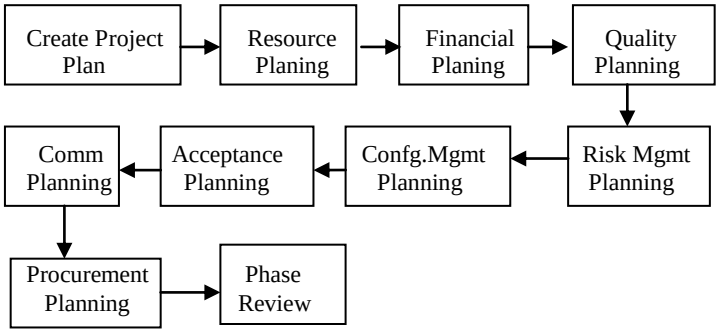
On the basis of the KPA actual suggestions and scope of improvement will be calculated



II. PLANNING PHASE

The **Planning Phase** is the second phase in the software life cycle. It involves creating of a set of plans to help guide your team through the execution and closure phases of the project.

The plans created during this phase will help you to manage time, cost, quality, risk and issues. They will also help you administer workforce and external suppliers, to make sure that you deliver the project on time and within budget.



(Basic objective of Planning Phase)

In our project we have emphasizes on all of the objective of this phase. Suggestion will be given on the scope of improvement.

Vision/Scope of Planning Phase

- Technology Validation Complete
- Functional Specifications Baselined [4]
- Master Project Plan Baselined
- Master Project Schedule Baselined
- Development/Test Environment Set Up

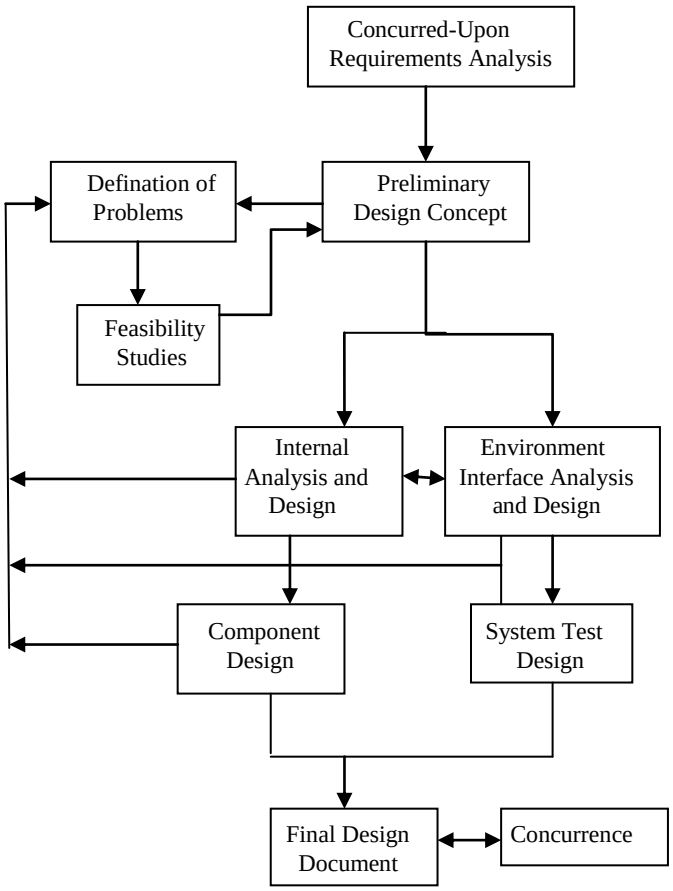
III. DESIGN PHASE

The Design phase is when you build the plan for how you will take your project through the rest of the SDL process—from implementation, to verification, to release. at some stage in Design phase you establish best practices to follow for this phase by way of functional and design specifications, and you carry out risk analysis to identify threats and vulnerabilities in your software.

During the Design Phase, the system is designed to satisfy the requirements identified in the earlier phases. The requirements identified in the Requirements Analysis Phase are transformed into a System Design Document that accurately describes the design of the system and that can be used as an input to system development in the next phase.

OBJECTIVES

- Transformation of all requirements into specifications covering all aspects of the system.
- Review and planning for security risks.
- Approval to growth of the Development Phase.



III. SECURITY PHASE

It is the most important phase of a SDLC. Your project must be reliable in order to obtain the trust of your users.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

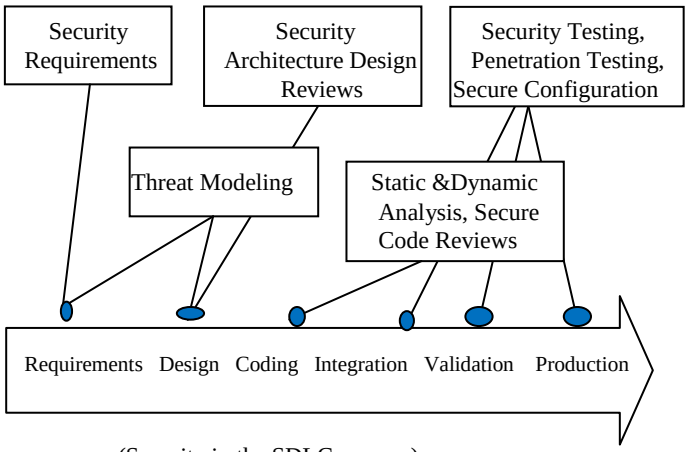
List of attacks that can harm a software

I. Surface Reduction Attack

Attack surface reduction is closely aligned with threat modelling, although it addresses security issues from a slightly different perspective. Attack surface decrease is a means of dropping risk by giving attackers less opportunity to exploit a potential weak spot or vulnerability. Attack surface fall encompasses shutting off or restricting access to system services, applying the principle of least benefit, and employing layered defines wherever possible.

II. Threat Modelling

Threat modelling is used in environments where there is meaningful security threat. It is a practice that allow development teams to consider, manuscript, and discuss the security implications of designs in the context of their planned operational environment and in a structured fashion. Threat modelling also allows consideration of security issues at the component or application level. Threat modelling is a team exercise, surrounding program/project managers, developers, and testers, and represents the primary security analysis task performed for the duration of the software design stage.



(Security in the SDLC process)

Use Approved Tools

All development teams should define and publish a list of approved tools and their associated security checks, such as compiler/linker options and warnings. This list should be accepted by the security advisor for the project team. In general, development teams should strive to use the latest version of approved tools to take advantage of new security analysis functionality and protections.

Deprecate Unsafe Functions

Many commonly used functions and APIs are not secure in the face of the current risk environment. Project teams should explore all functions and APIs that will be used in conjunction with a software development project and prohibit those that are determined to be unsafe. Once the barred list is determined, project teams should use header files (such as banned.h and strsafe.h), latest compilers, or code scanning tools to check code (including legacy code where appropriate) for the existence of barred functions, and replace those barred functions with safer alternatives.

Static Analysis

Project teams should perform static analysis of source code. Static investigation of source code provides a scalable capability for security code review and can help ensure that secure coding policies are being followed. Static code investigation by itself is generally insufficient to replace a manual code examination. The security team and security advisors should be aware of the strengths and weaknesses of static analysis tools and be prepared to augment static analysis tools with other tools or human review as appropriate.

IV. IMPLEMENTATION

The Implementation phase is when the end user of your software is foremost in your mind. In this phase you make the documentation and tools the customer uses to make informed decisions about how to deploy your software securely. To this end, the Implementation phase is when you establish development best practices to detect and remove security and privacy issues early in the development cycle.

OBJECTIVES

- System deployment
- Training on the system

GOALS

The purpose of the Implementation Phase should is to deploy and enable operations of the new information system in the production environment.

ROLES

The following personnel contribute in the work activities during this phase:

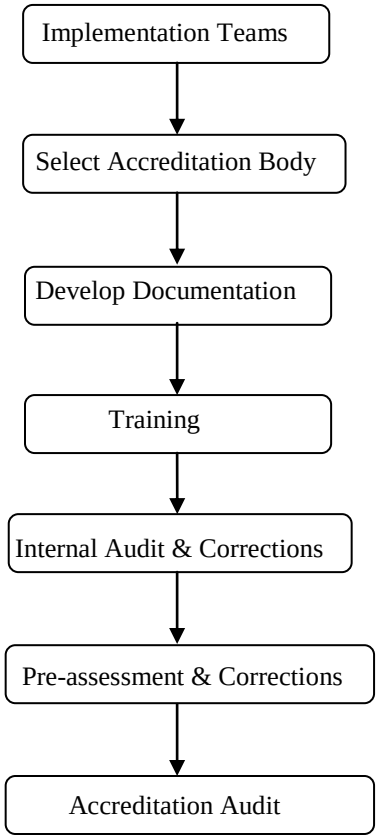
- Agency CIO
- Project Sponsor
- Executive Sponsor
- Project Manager
- Development Team

The major steps involved in this phase are:

- Acyuisition and Installation of Hardware and Software
- Conversion
- User Training
- Documentation

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015



(Implementation Phase)

V. CONCLUSION

Thus, in this paper we have defined and explained the software process improvement model which consists of the three basic phases i.e. planning, design, security, implementation. In this model we have calculated the KPA on the basis of quality questions and the suggestions for changes if any are given, taking out the weak areas that needs attention. The overall result of each phase is depicted in the form of graph.

VI. FUTURE WORK

In future we will be working on the evaluation of codes in a much optimized way. We will integrate a compiler which will scan the fragment of codes and on the basis of its analysis we will generate a log file which will be indicating the areas of improvement. Assessment will be done on the basis of time and space complexity.

REFERENCES

1) Mamta Shelpar, Sona Malhotra,"Software Process Improvement Model", *Iternational Journal of Advanced Research in Computer Science and Software Engineering*, June 2013.
2) D.F.Rico, "ROI of Software Process Improvement". *J.Ross Publication* 2004.
3) Ministry of Communications and Information Technology, "Software Process Improvement Standards& Specification" *NIC/SD/SPISS V1.0:2009*.
4) Ministry of Communications and Information Technology, "Software Process Improvement Standards& Specification" *NIC/SD/SPISS V1.0:2009*.
5) Ruth Klendauer, Axl Hoffiman, Jan Macro Leimeister and Marina Berkovich, Helmut Krcmar, "Using the IDEAL Software Process Improvement Model for Implementation of Automotive SPICE," *2012 IEEE CHASE 2012, Zuric*.

.