Arab J Sci Eng (2016) 41:997–1014
DOI 10.1007/s13369-015-1933-1

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

# Design and Application of New Quality Improvement Model: Kano Lean Six Sigma for Software Maintenance Project

**G. ArunKumar**[1] · **R. Dillibabu**[1]

Received: 19 June 2015 / Accepted: 13 October 2015 / Published online: 18 November 2015


**Abstract** The continual changes in requirements impact the expectation about the quality of the product as well as the process, namely software project. In this paper, the various limitations that still exist in the software development process are presented. It is aimed to develop a new quality improvement model to enhance software quality without increasing effort, cost, and time. To achieve a software project of expected quality, a new quality improvement model, namely Kano Lean Six Sigma model (KLSS), is proposed. The KLSS model is used to identify the exact requirements for the software project from the customer's perspective. KLSS helps to categorize the requirements based on the nature of the defect, to eliminate the requirements of non-value processes and to implement the main functionality to meet the expectations of the customer. As regards our proposed software maintenance project, the method of development has been suitably tested in a leading IT company. The model has shown greater improvement in quality, cost, and efforts.

**Keywords** Software requirements · Kano · Lean Six Sigma · Software development · Quality improvement model

## 1 Introduction

Software development process provides an important method for organizations to rapidly apply knowledge within key strategic and operational processes. The process allows the firm to use present knowledge through incremental changes to existing systems (e.g., maintenance and support activities) besides enabling the exploration of new avenues for action through the creation of new systems (e.g., new software project initiatives). A well-defined software development process can provide an effective product, thereby paving the way for achieving long-term success in the market [1]. In contrast, an ineffective software development process can hamper the development and utilization of organizational knowledge and prevent timely responses to market changes [2]. The software development process fails due to a number of reasons [3]. Some of the issues are mentioned below:

- Identifying various requirements and noting changes taking place over a period of time.
- The requirements specified not being clear enough for the development phase.
- Each phase of the software development life cycle having a testing process.
- Ensuring that every feature/characteristic is computed before proceeding to the next one.
- Involving the customer in the entire development process.
- The support being limited for distributed development environments and involving large teams.

To gain a comprehensive understanding of customer's requirements and satisfaction levels, Kano et al. [4] proposed a two-dimensional quality model (Kano's model) to create the relationships between quality attributes and customer satisfaction, by modifying the two-dimension quality model developed from the dual factor theory proposed by Herzberg et al. [5]. The modified model can effectively evaluate and improve quality performance or develop new products and services to satisfy the needs and expectations of customers

✉ G. ArunKumar
aarunkumar81@gmail.com

R. Dillibabu
dillibabu@annauniv.edu

[1] Department of Industrial Engineering, Anna University, Chennai 600025, India

[6–8]. Hence, Kano's model has been widely applied in various fields [6–14] and proved as a considerably useful tool for quality attribute classification and customer satisfaction enhancement. Lee et al. [7], Lee and Huang [8], and Hu et al. [15] proposed a quantitative analysis model of Kano's model to make improvement analysis and decision making more accurate. With the Kano's model, we can determine the customer satisfaction levels. It also helps categorize the customer requirements based on needs, providing a solution and better understanding of their needs and expectations.

More recently, Lean Six Sigma has become more popular as organizations strive to meet the quality objectives defined by their customers. However, Lean Thinking emerged from the Japanese automobile and started receiving attention in the USA and Western Europe in the 1980s. Similarly, Six Sigma was introduced in the 1980s by Motorola. However, this concept is the result of a series of developments in quality management that started in the early 1930s. The importance of combining the strength of two approaches was highlighted by Hoerl [16] and Antony et al. [17], and they discussed the theoretical possibility of integrating these two approaches. Arnheiter and Maleyeff [18] made it clear that when they used the term Lean Six Sigma, they referred to an integrated entity. By combining Lean Thinking and Six Sigma Ricondo and Viles [19] noticed a considerable change and argued that the variability reduction focus of Six Sigma enhanced the robustness of fragile Lean systems. They identified some natural conflicts while applying some methodologies to the existing system but at the same time he was attracted toward the approach of integrating Lean and Six Sigma advantages.

The experiences of de Koning et al. [20] saw the successful implementation of Lean Six Sigma in the organizational infrastructure, by applying the idea within the project framework. Byrne et al. [21] agreed that Lean and Six Sigma balanced each other and characterized a powerful union that eliminated process waste and variation in a process. Carleysmith et al. [22] narrated his experiences of the implementation of Lean Thinking and Six Sigma in pharmaceutical research and development (R&D). He successfully improved the process of new pharmaceutical manufacturing. Perrer et al. [23] examined the integration of Lean principles with Six Sigma methodology as a logical approach to continuous improvement and provided a conceptual model for their successful integration. Gamal Aboelmaged's et al. [24] study clearly showed that only explicit barriers considerably influenced Six Sigma implementation in relation to dimensions of organizational factors. Vinodh et al. [25] and Vinodh and Prasanna [26] identified that the Lean anchorage in the LSS process was rather weak and that it had to be enhanced for improving the effectiveness of LSS approach. Later, Assarlind et al. [27] gave a theoretical foundation for LSS by studying and analyzing its practical applications. As a result, it gave a new factor of significance for successful Lean Six Sigma applications, such as having a clear organization that guides the company in terms of what mechanism of Lean Six Sigma to be applied and what competences to be involved in various projects depending on its capacity and complexity. Thus, LSS methodologies serve to improve processes, eliminate product or process defects, reduce cycle times, and accelerate processes.

This paper aims to overcome the issues in software development process by combining the models of Kano and Lean Six Sigma to develop a suitable new model. The Kano Lean Six Sigma Model (KLSS) will have the methodology and tools for identifying the customer requirements. It will categorize customer requirements based on the nature of the defects and will eliminate unwanted process and requirements. Finally, it will reduce the variation in the process, thereby increasing the speed of project delivery.

## 2 Related Works

This study focuses on software development by applying the Kano model during the requirements analysis phase and to verify the feasibility of applying Lean and Six Sigma for software development and for software application projects. Kano et al. [4] investigated the quality in two aspects such as subjective and objective. So it proposed the two-dimensional recognition system which replaced the one-dimensional system to categorize and identify the customer attributes. Xu et al. [28] proposed a method used to analyze the customer satisfaction. It supported two methods for decision-based product design. The product design analyzed the customer needs, and configuration index made decision factor of product configuration design. So, it was used to visualize the impact of the requirements on customer satisfaction. Sharma et al. [29] integrated the Lean principle with Six Sigma practices which used a different number of methods used to reduce the cost of the manufacturing process. This process improved the customer satisfaction level in battery industries. Furterer and Elshennawy [30] presented the TQM-based Lean Six Sigma approach for applying to the local government to improve the quality, decrease the variation, and remove the waste from the organization. So, it proposed continuous process improvement in the public finance department. Marti [31] applied Lean and Six Sigma process using the data and statistical analyses to measure the quality and improvement of the organization by identifying and removing the threats in the process. This paper applies the Lean and Six Sigma process to the Pharmaceutical Industry for improving the quality by focusing on the customer needs. Chao et al. [32] developed the integrated Lean Six Sigma methodology for improving the quality of the services. This process focused on satisfying the customer requirements and produced the linear process output in the different service industry. Kumar et

al. [33] used Lean Six Sigma process methodology to eliminate final product errors. The lean tool combines with Six Sigma DAMIC methodology for achieving the customer's fidelity and improving the bottom line result. D'Angelo and Zarbo [34] implemented the method for achieving the zero defect performance goal. For increasing the quality and the performance, a novel data collection was used to achieve continuous improvement in the service industry. Tonini et al. [35] introduced the Lean with Six Sigma process for achieving the error-free products. This approach helped to identify the project mistakes easily so that the projects could be finished more quickly and effective results could be got. Kanakana et al. [36] applied the Lean Six Sigma method to the engineering education for improving the throughput of the students and faculty revenue. From the survey, it becomes clear that Kano model could be integrated with any model for identifying and prioritizing customer requirements. Xu et al. [28], Mcllroy and Silverstein, and Sharma [29] showed that Lean Six Sigma could be implemented in the non-manufacturing and service industries and showed a significant improvement. Likewise, implementation of Lean Six Sigma was initiated in software development process by Tonini et al. [35]. Later, a software development approach was created using Lean Six Sigma to hold the continuity and change requirements by Pillai et al. [37,38].

The following section describes the proposed Kano Lean Six Sigma model construction procedure, implementation details, and the related result analysis.

## 3 Construction of KLSS Model

The KLSS has been developed based on two important powerful integration models, namely Kano and Lean Six Sigma. The integration of these models has been done in a stagewise manner. The validation process of the model has also been built in a stagewise fashion. The framework of KLSS is shown in Fig. 1. The following are the steps for the development of KLSS methodology:

Step 1. Get the product requirements from the customers with all its functionalities and features using Kano model.
Step 2. Categorize the requirements and prioritize them based on the expectation of the customer using Kano model.
Step 3. Document the requirements of the product and the process of the software project clearly.
Step 4. Identify the critical-to-quality (CTQ) in the software development phases and eliminate the unwanted process and implement the main functionality in the process.
Step 5. Calculate the risk priority number (RPN) and check whether any specific changes in the process have their causes and desired effects.
Step 6. Document the issue solving approach and product information using thought process mapping (TPM).
Step 7. Create a process map for the new process to know how the process works today.
Step 8. Perform Pareto Chart analysis and brain storming techniques to encourage creativity
Step 9. Identify non-value-added process using failure mode effective analysis (FMEA)
Step 10. Compute the risk priority number (RPN) using the formula

$$\text{RPN} = \text{Severity} \times \text{Occurrence} \times \text{Detection}$$

Step 11. Eliminate time delays in the process using thought process mapping (TPM)
Step 12. Check if the specific changes in the process have desired effects and their causes
Step 13. Finalize the Process Improvement Method

Initially, the KLSS model begins with the getting of the requirements from the user, and based on the expectation from customer it will prioritize the requirements using Kano model. During LSS implementation process, it will identify the CTQ in the requirements. Then, the risk involved and its desired effects will be assessed, and finally through brainstorm sessions and thought process mapping, all issues will be sorted out with the customers. A process map is created for the new process where the new process should eliminate the waste process or non-value process in the current process (e.g., requirements already implemented or solved) and the failure mode effective analysis (FMEA) is performed and the RPN value is calculated for the current process. By comparing the results of RPN values before and after applying KLSS model, changes, if any, are found and if there is a big change in the process, the current process is made the standard one and the new process improvement model is followed. These steps which will be implemented in an application development project are listed below.

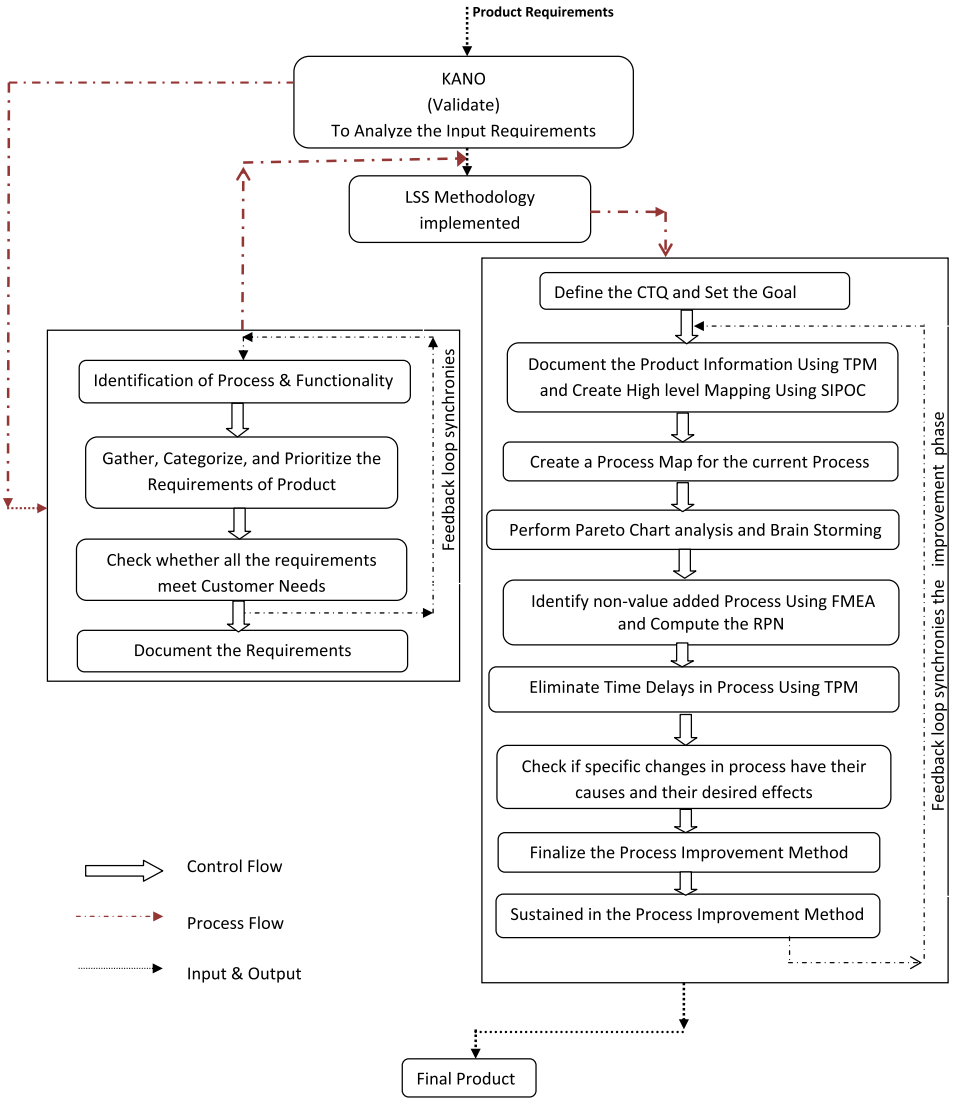
## 4 Application and Implementation of KLSS

A case study was conducted in a leading software company in India. The company is CMM level-5 certified in developing software projects for embedded software applications. It uses the online review process for software quality improvement.

This project is basically used for upgrading the embedded hardware and software functionality of devices in photocopier machine. The Software Upgrade can be performed with the functions listed below:

(i) Power On upgrade
(ii) Manual network upgrade

**Fig. 1** Proposed model KLSS




(iii) Automatic network upgrade
(iv) USB pen drive upgrade
(v) USB pen drive Altboot
(vi) PWS upgrade
(vii) Scanner copier software upgrade
(viii) Portfolio system release (PSR) upgrade

The following are the implementation details:

(i) The Kano model was used to prioritize the requirements based on the ranking in the customer's expectation and time of delivery of requirements (see Table 1; Fig. 2).
(ii) Thought process map of Scanner copier software upgrade can be plotted, and it provides a visual map that tracks the development of ideas and issues as well as the extent of inquisition.
(iii) The existing process is mapped using process map As-Is, and the estimated time for the existing process is calculated based on the defects or issues Per Kilo Lines of Code(KLOC) (see Fig. 3).
(iv) Critical-to-quality (CTQ) details are listed in the existing project which reveals information to code developers and external code reviewer about defects or issues to be overcome using KLSS model (see Table 2).
(v) The KLSS model includes occurrence rating, severity rating, and detecting ability rating table for finding the defect criteria (see Table 3).
(vi) The KLSS model is implemented and remedial measures in developing process are carried out with the help of FMEA and the process map for 'Should-Be plotted' of new process, thereby validating KLSS model using the variable RPN which shows that there is a significance in terms of quality, time, and effort in the implementation of KLSS model (see Tables 4, 5; Figs. 4, 5).
(vii) Finally, the new controlling process is identified with various new methods, frequency of audit, and recom-

**Table 1** Prioritization of customer needs

| S. No. | Requirements | Importance ranking |
|---|---|---|
| Q.1 | Automatic network upgrade | 7.4 |
| Q.2 | Power ON software upgrade | 8.5 |
| Q.3 | Manual network upgrade | 7.0 |
| Q.4 | USB Pen drive upgrade | 6.5 |
| Q.5 | USB Pen drive Altboot | 5.3 |
| Q.6 | PWS upgrade | 5.7 |
| Q.7 | Portfolio system release upgrade | 4.6 |
| Q.8 | DML filenames | 6.4 |
| Q.9 | Software compatibility database | 5.3 |
| Q.10 | DML files with and without embedded pages | 3.4 |
| Q.11 | Network controller and network controller OS upgrade | 3.4 |
| Q.12 | IIT And DADH | 4.4 |
| Q.13 | Common interface upgrade | 5.0 |
| Q.14 | Finisher module upgrade | 6.0 |
| Q.15 | Multitech modem firmware | 4.9 |
| Q.16 | Color control management | 4.4 |
| Q.17 | KMIOT application | 4.3 |
| Q.18 | General software upgrade | 4.1 |

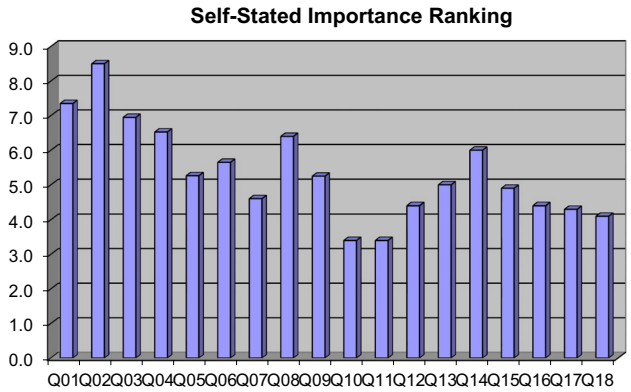



**Fig. 2** Importance ranking of customer needs

mended actions being identified for this process which should be followed after implementation of KLSS model (See Table 6).

### 4.1 Kano Model for Getting Customer Requirements

The Kano model was applied for getting customer requirements and to improve the quality of project delivery. The following are the implemented details:

(i) The Kano model is used to get the voice of customer requirements during the maintenance of the project; it addresses all the requirements from different perspectives.
(ii) It will categorize the requirements in different quality attributes as Attractive quality attributes requirements (A),One-dimensional quality attributes Requirements (O), Must Be quality attributes (M), Indifferent quality attributes requirements (I), and Reverse quality attributes requirement (R).
(iii) For each requirement, weight will be assigned based on the quality attribute, and it will be prioritized and importance ranking will be calculated.
(iv) Customer satisfaction and dissatisfaction level will be calculated.
(v) Finally, the customer importance requirements will be ranked and requirements with high priority will be implemented first.
(vi) Then, the least priority requirements will be implemented in the next iteration of implementation.

Calculation of Kano model is performed based on the following methods for various requirements, and importance ranking is identified for embedded software applications (see Table 1).

From Fig. 2, it is identified that requirements no Q.2, Q.1, and Q.3 have high priority and need to be addressed first by the developer and the importance of the requirements has been ranked, which makes the developer fix the issue in the first iteration itself.

### 4.2 Mapping Ideas and Innovation Using the Thought Process Map (TPM/TMAP)

A visual representation of team's thoughts, ideas, and questions pertaining to the accomplishment of the project goal is necessary. It is an effective tool for ensuring that all potential questions and issues of the project have been both identified and addressed from the beginning of the project to its completion. It also provides an effective way to brainstorm, take notes, gather and view information, and summarize the data. It reminds the team of the assumptions made, the actions that followed, and the latest status of the project. It is an effective way of communicating as well as consolidating information from a single person or among various teams. Finally, it provides a visual map that tracks the development of ideas and issues as well as the extent of inquisition. It helps to find the solution for the problem in a much easier way, and it takes the problem to the threshold of the solution by getting different ideas and thoughts of different customers. TPM will be helpful in getting a clear idea of the process of finding what the customer is expecting in the end product.

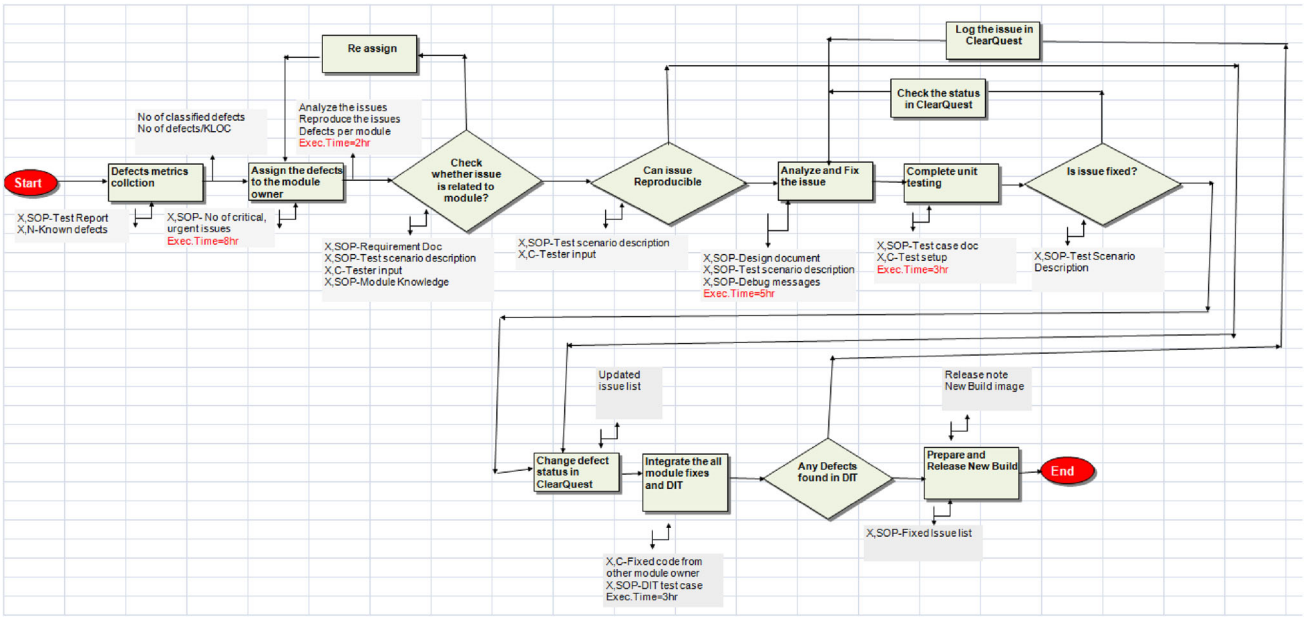



**Fig. 3** Process map As-Is for existing process

### 4.3 Process Map As-Is for Existing Process

Currently it depicts a process; "As is" process maps are usually characterized by several input options, bottlenecks, multiple handoffs, inspections, and rework loops. It is the starting point to the understanding of how a process runs. It becomes a "Should Be" map once all non-value-added activities have been removed from the "As Is" process, after careful analysis. For the existing software, the implementation process has been drawn as shown in Fig. 3 and the steps are listed below:

Step 1. Collect the defect metric from the customer.
Step 2. Give clear quest status not completed to the defect of the module owner.
Step 3. Otherwise reassign the defect to the next module owner.
Step 4. Check whether the issues are related to the same module or a different one.
Step 5. Analyze and fix the issue and perform the complete unit test.
Step 6. Check the status in clear quest and find if the issue solved changed the defect status in clear quest as completed.
Step 7. Integrate all modules and fix the defect in development integration testing (DIT).
Step 8. Otherwise prepare and release new built product.
Step 9. End the development process.

Before performing the operation, we need to identify the number of classified defects and the number of defects per KLOC and we need to identify the test report and also we need to identify the number of critical or urgent issues where we need to make an analysis of issues whether we are going to reproduce the issues or they have been misjudged by the customer but actually implemented.

If a defect or issue is going to be reproduced, the following information is required for implementing the module: they are requirement document, test scenario description, tester inputs, test case, fixed issue list, and module knowledge. In the existing process, for all the defects or issues, all the above operations need to be performed; this results in waste of time and man-hours. A sample critical-to-quality defect list is shown below in Table 2 and the man-hours for the current process for identifying and eliminating the issues and defects have been calculated.

### 4.4 Failure Mode Effective Analysis

FMEA is one of the first systematic techniques for failure analysis. An FMEA is often the first step of a system reliability study. It involves reviewing as many components, assemblies, and subsystems as possible to identify failure modes, and their causes and effects. For each component, the failure modes and their resulting effects on the rest of the system are recorded in a specific FMEA worksheet. FMEA also helps to identify the value and non-value-added process

**Table 2** Critical-to-quality list for the scan copier module

| S. No. | Program | Headline | Change type | Priority | Submit date | Assigned date | State | Resolved date | Work status | Software module |
|---|---|---|---|---|---|---|---|---|---|---|
| CQ1 | Scan copier | Scanner is not working after upgrading 15000 and 14200 via Netboot and Altboot | SwDefect | P2-Fix for Launch | 04-Jun-12 | 07-Jun-12 | Closed | 15-Jun-12 | Replication | CCS_SWUPGRADE |
| CQ2 | Scan copier | DADH BlockStatusUpdate and BlockFailed messages are not forwarded by IIT to IIT-GUI/SWUG | SwDefect | P2-Fix for Launch | 04-Jun-12 | 07-Jun-12 | Closed | 15-Jun-12 | Investigation/ analysis | CCS_SWUPGRADE |
| CQ3 | Scan copier | Software upgrade crash with extra feeders | SwDefect | P2-Fix for Launch | 04-Jun-12 | 07-Jun-12 | Closed | 15-Jun-12 | Fixing | CCS_SWUPGRADE |
| CQ4 | Scan copier | SW upgrade should be modified to not downgrade IME SW [Clone from CQGbl00000] [Clone from CQGbl0000000] | SwDefect | New Require-ment | 04-Jun-12 | 07-Jun-12 | Resolved | 10-Jun-12 | Build creation | CCS_SWUPGRADE |
| CQ5 | Scan copier | Software compatibility check does not run after a software upgrade if the FAX is not installed or Disabled | SwDefect | P2-Fix for Launch | 08-Jun-12 | 08-Jun-12 | Resolved | 15-Jun-12 | Test | CCS_SWUPGRADE |
| CQ6 | Scan copier | [PROGRAM GOAL] A method is required to allow a Tesla launch Clone file onto Tesla Plus | SwDefect | P2-Fix for Launch | 17-Oct-12 | 26-Jun-12 | Resolved | 26-Sep-12 | Delivering | CCS_SWUPGRADE |
| CQ7 | Scan copier | Spyglass DPAT software upgrade to .18810 failed {active} | SwDefect | P1-Critical Barrier | 10-Jul-12 | 12-Jul-12 | Closed | 16-Jul-12 | Requested for retest | CCS_SWUPGRADE |
| CQ8 | Scan copier | When fitting a Radiance 2K LCSS to a Luminance the fin s/w upgraded and the Scan Engine. | SwDefect | P2-Fix for Launch | 10-Jul-12 | 12-Jul-12 | Closed | 16-Jul-12 | Barrier | CCS_SWUPGRADE |
| CQ9 | Scan copier | After normal SW upgrade we will see more than 2 reboots before machine is available {inactive} | SwDefect | P1-Critical Barrier | 08-Jun-12 | 26-Jun-12 | Closed | 16-Jul-12 | External depen-dency | CCS_SWUPGRADE |
| CQ10 | Scan copier | Software upgrade of scan engine failed- progress bar stopped 2/3 way across | SwDefect | P2-Fix for Launch | 12-Jul-12 | 14-Aug-12 | Closed | 13-Sep-12 | Code review | CCS_SWUPGRADE |

**Table 2** Countinued...

| S. No. | Program | Headline | Change type | Priority | Submit date | Assigned date | State | Resolved date | Work status | Software module |
|---|---|---|---|---|---|---|---|---|---|---|
| CQ11 | Scan copier | {moredata} Device resets after software upgrade | SwDefect | P2-Fix for Launch | 01-Aug-12 | 14-Aug-12 | Closed | 02-Aug-12 | Code review | CCS_SWUPGRADE |
| CQ12 | Scan copier | Code on scanners on mamba plus printers may get corrupted | SwDefect | P1-Critical Barrier | 29-Mar-12 | 14-Aug-12 | Resolved | 12-Sep-12 | Test | CCS_SWUPGRADE |
| CQ13 | Scan copier | SOFTWARE UPGRADE - Add upgrade support XOM spyglass releases | SwDefect | P2-Fix for Launch | 18-Jun-12 | 06-Jul-12 | Resolved | 16-Jul-12 | Test | CCS_SWUPGRADE |
| CQ14 | Scan copier | Machine reports a scanner fault when software upgrade mode is requested | SwDefect | P2-Fix for Launch | 08-Jun-12 | 26-Jun-12 | Closed | 13-Sep-12 | Investigation/ analysis | CCS_SWUPGRADE |
| CQ15 | Scan copier | {inactive} SRT-Due to Error code 67686504, the upgradation of the device to .236 Failed in devices | SwDefect | P2-Fix for Launch | 30-Aug-12 | 04-Sep-12 | Closed | 13-Sep-12 | Investigation/ analysis | CCS_SWUPGRADE |
| CQ16 | Scan copier | SECURITY: In luminance, manual software upgrade is getting passed when using same software version | SwDefect | P1-Critical Barrier | 01-Aug-12 | 14-Aug-12 | Closed | 13-Sep-12 | Build creation | CCS_SWUPGRADE |
| CQ17 | Scan copier | When fitting a radiance LVF to a luminance the fin s/w upgraded and the scan engine | SwDefect | New Require-ment | 08-Jun-12 | 26-Jun-12 | Closed | 13-Sep-12 | Code review | CCS_SWUPGRADE |
| CQ18 | Scan copier | 30 minute SW upgrade timer expires before the upgrade has completed | SwDefect | P2-Fix for Launch | 04-Sep-12 | 04-Sep-12 | Resolved | 12-Sep-12 | Fixing | CCS_SWUPGRADE |
| CQ19 | Scan copier | Scan engine upgrade stuck-up during Altboot | SwDefect | P1-Critical Barrier | 04-Sep-12 | 04-Sep-12 | Closed | 13-Sep-12 | Code review | CCS_SWUPGRADE |
| CQ20 | Scan copier | [Sps n/a; Jps01 26/13] {active} SRT-“Reset the Device” after attempting SWUP on a Javelin device from .236 version to .248 version via Web UI method | SwDefect | P2-Fix for Launch | 26-Jun-12 | 26-Jun-12 | Closed | 16-Jul-12 | Code review | CCS_SWUPGRADE |

**Table 3** Severity, occurrences, detectability rating process FMEA

| Effect | Rating | Criteria |
|---|---|---|
| *Severity rating process FMEA* | | |
| Very high | 10 | Resulting rework is greater than 8 man-hours |
| High | 7 | Resulting rework is greater than 4.0 person-hour and less than 8.0 man-hour |
| Moderate | 5 | Resulting rework is greater than 2.0 person-hour and less than 4.0 man-hour |
| Low | 3 | Resulting rework is greater than 1.0 person-hour and less than 2.0 man-hour |
| Very Low | 1 | Resulting rework is less than 1.0 man-hour |
| *Occurrences rating process FMEA* | | |
| Very frequently | 10 | If the process data analysis shows greater than 80 % of defects attributable to the sub- process failure |
| Frequently | 8 | If the process data analysis shows greater than 40 % of defects and less than 80 % of defects attributable to the sub-process failure |
| Sometimes | 4 | If the process data analysis shows greater than 10 % of defects and less than 40 % of defects attributable to the sub-process failure |
| Rare | 1 | If the process data analysis shows less than 10 % of defects attributable to the sub-process failure |
| Detection | Rating | Criteria |
| *Detectability rating process FMEA* | | |
| Difficult | *10* | No defined methods for identifying the process error; only the output product analysis leads to detectability |
| Moderate | *5* | Can be identified in the exit phase of the process or subsequent process entry check |
| Easy | *1* | Process has built in checks for identifying subprocess failure |

steps through which we can find the root causes for the existing process inefficiency. This will identify and eliminate the risk factors of each process to avoid/reduce failure. For the current process of software implementation, FMEA has been plotted in the work sheet. FMEA model includes occurrence rating, severity rating, and detecting ability rating. Table 3 can be used for finding the defects criteria for the current process and risk priority number being calculated.

### 4.5 Process Map Should-Be for Current Process

A depiction of a new and improved version of a process where all non-value-added steps have been removed is based on:

- Everything being done right the first time
- Customer requirements built into the process
- Flexibility to meet multiple customer types or requirements
- Design with "process" versus "functional" mindset
- Limited handoffs and inspections
- Easy to document, manage, train, and control
- Several possible inputs
- Bottlenecks eliminated
- Handoffs, inspection, and rework loops no longer needed

The following steps are followed in the process map Should-Be for the current software implementation of embedded software applications process which leads to an improvement in software implementation approach, thereby reducing the time and work man-hours. The process map Should-Be for the new process of software implementation is shown in Fig. 4 and the implementation steps are listed below:

Step 1. Collect the defect metrics or issues from the customer.
Step 2. Classify the issues based on the priority and severity of defects.
Step 3. Review the defects and check if they are valid or not; if valid, continue the process; If not, close the defect.
Step 4. If the defect is valid, assign defect or issue to module owner.
Step 5. Check the nature of defect or issue, i.e., whether it is new defect or reproducible defect.

**Table 4** RPN 1 and RPN 2 calculation using FMEA

| # | Process step or part | Potential failure modes (FM) | Potential failure effects | Severity (S) | Potential causes | Occurrence (O) | Current controls | Detection (D) | RPN 1 | Actions recommended | Severity (S) | Occurrence (O) | Detection (D) | RPN 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Prepare test case document** | No proper understanding of the system | Number of invalid issues increases | 7 | Insufficient knowledge or negligence | 8 | Requirement document | 3 | 168 | Review of test cases with the partner along with the development | 7 | 4 | 3 | 84 |
| 2 | **Developer integration testing** | Not covering all the scenarios, no regressive testing | Number of issues increases | 10 | Lack of time | 4 | Proper estimation for DIT | 3 | 120 | Allocating time for DIT and DIT issues fixing and verification | 5 | 4 | 3 | 60 |
| | | | | | Lack of understanding the use cases | | Avoiding final minute changes | | | | | | | |
| | | | | | Freezing the code | | Understanding the use cases | | | | | | | |
| 3 | **Improper fixes/work-around** | Lack of understanding the cause of the issues | Number of issues increases related to this workaround | 10 | Improper issue allocation | 2 | Reallocation of issues | 7 | 140 | Root cause analysis and code review | 7 | 1 | 2 | 14 |
| 4 | **Reproduce the issue** | Improper procedure description | Delay in reproducing the issue | 5 | Detailed procedure is missing | 7 | Procedure description | 3 | 105 | Define the detailed procedure to reproduce the issue | 5 | 4 | 3 | 60 |

**Table 4** Countinued...

| # | Process step or part | Potential failure modes (FM) | Potential failure effects | Severity (S) | Potential causes | Occurrence (O) | Current controls | Detection (D) | RPN 1 | Actions recommended | Severity (S) | Occurrence (O) | Detection (D) | RPN 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | **Availability of hardware component** | Delay in finalizing the supported devices and not getting the supported device on time | Prolonged testing time and high utilization of resources | 10 | Delayed delivery in getting the supported devices | 6 | Follow-up of the ordered devices | 1 | 60 | Get the hardware delivered while freezing the requirement | 5 | 1 | 1 | 5 |
| 6 | **Assigning the defect to owner** | Improper domain knowledge | Delay in defect analysis | 7 | Lack of domain knowledge | 2 | Avoid inefficient defect assignition | 4 | 56 | Confirm the defect assignition with experts | 7 | 1 | 4 | 28 |
| 7 | **Requirement analysis** | No proper understanding of the requirements | Customer satisfaction rating decreases | 10 | Lack of requirement understanding | 4 | Requirement document | 1 | 40 | Confirm the requirement understanding regarding the module work | 7 | 1 | 1 | 7 |
| 8 | **Legacy issues** | Lack of legacy system understanding | Number of legacy issues increases | 1 | Unavailability of legacy issues document | 5 | Fixing the urgent issues | 5 | 25 | Document the legacy issues and considering during development | 1 | 1 | 1 | 1 |
| 9 | **Unit test** | Unit test code failed | Lead to failure of actual functionality | 7 | Lack of execution of unit test code | 3 | Unit test code not executed | 1 | 21 | Need to execute the unit code before deliver the code | 1 | 1 | 1 | 1 |
| 10 | **Missing caveats** | Preconditions are missing | chances of ambiguity | 1 | Ambiguity | 3 | Precondition not defined | 6 | 18 | Define the precondition | 1 | 1 | 1 | 1 |

**Table 5** Results of FMEA of RPN1 and RPN2

| S. no. | Causes | RPN 1 | Cumm RPN | % Cumm RPN | RPN 2 | Cumm RPN | % Cumm RPN |
|---|---|---|---|---|---|---|---|
| 1 | Prepare test case document | 168 | 168 | 22 | 84 | 84 | 11 |
| 2 | Improper fixes/workaround | 140 | 308 | 41 | 60 | 144 | 55 |
| 3 | Developer integration testing | 120 | 428 | 57 | 14 | 158 | 61 |
| 4 | Assigning the defect to owner | 105 | 533 | 71 | 60 | 218 | 84 |
| 5 | Availability of hardware component | 60 | 593 | 79 | 5 | 223 | 85 |
| 6 | Reproduce the issue | 56 | 649 | 86 | 28 | 251 | 96 |
| 7 | Requirement analysis | 40 | 689 | 92 | 7 | 258 | 99 |
| 8 | Legacy issues | 25 | 714 | 95 | 1 | 259 | 99 |
| 9 | Unit test | 21 | 735 | 98 | 1 | 260 | 100 |
| 10 | Missing caveats | 18 | 753 | 100 | 1 | 261 | 100 |
| | Total | 753 | | | 261 | | |

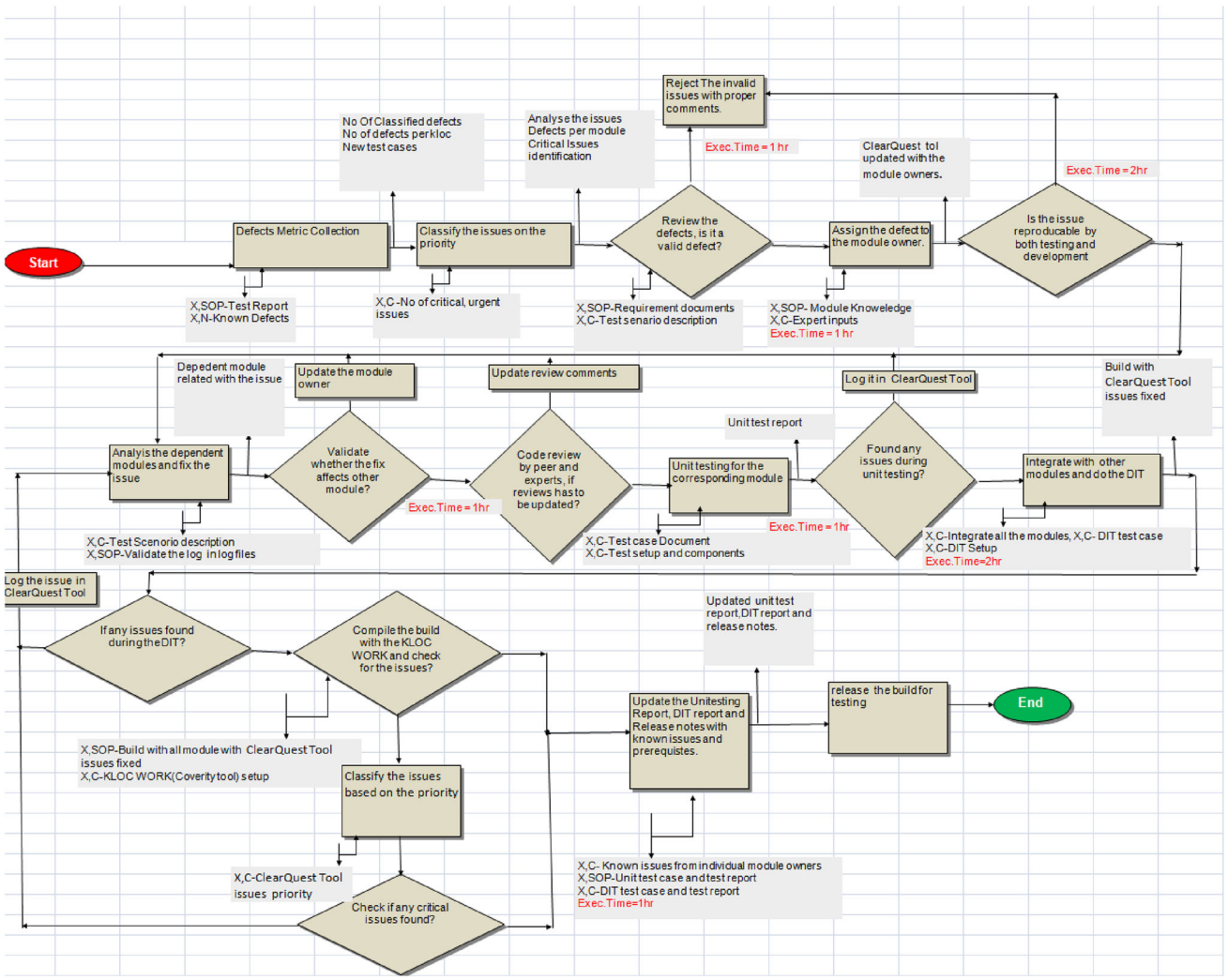



**Fig. 4** Process map Should-Be for the new process

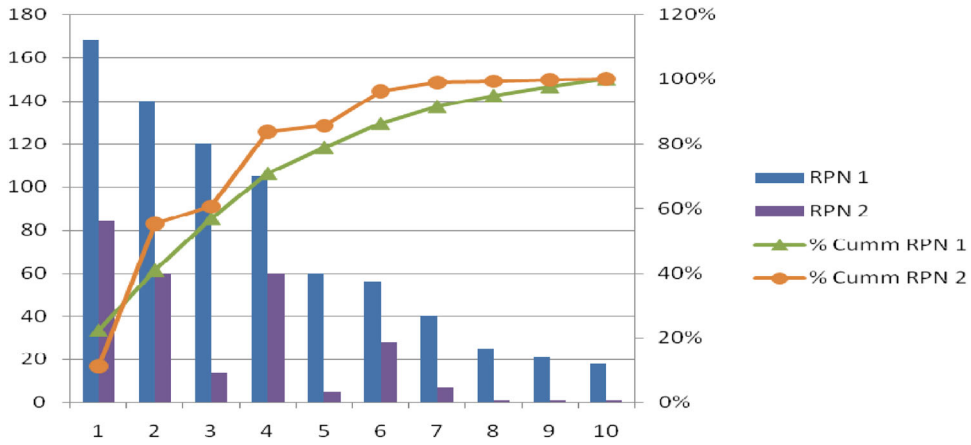



**Fig. 5** Pareto chart for the RPN values (RPN 1): existing process value RPN 2: new process improvement value)

Step 6. Analyze the dependent module and fix the issue
Step 7. Validate the module and check whether the fixed defect will affect other modules.
Step 8. Perform code review process by peers and experts and the review reports to be updated.
Step 9. Perform a unit test for the corresponding module.
Step 10. Integrate the module and perform the DIT.
Step 11. If any issue is found during the DIT, fix it.
Step 12. Update the unit test report, DIT report, release the note with all the known issues.
Step 13. Release the built product after testing.

Figure 4 shows that the new process is capable of addressing all the issues or defects in the initial level of the process and later shows that a large volume of non-value-added process has been identified and discarded in the initial level itself. Therefore, a huge amount of work man-hours has been reduced. Meanwhile, it leads to faster delivery of the issues or defects being rectified, thereby preventing the quality of the project from being affected.

Based on the FMEA, it is easy to identify the potential modes of failures. Its effects can be visualized through this severity, occurrences, and detection being rated for each failure, and the RPN values (risk priority number) are calculated for current process and in the same way the RPN is calculated for the new process, with all data being recorded in the work sheet. Finally, the Pareto chart is drawn between RPN 1 and RPN 2 to identify any changes or improvement in the current process. From a comparison of the results of RPN1 and RPN2 (Fig. 5), it becomes clear that the RPN 2 process is more efficient and effective in handling the defects in the current process. The total time saved as a result of this process (i.e., improvement) is 1399 man-hours.

| | |
|---|---|
| *Man-hours calculation:* | |
| Project Total KLOC | = 45 |
| Man-hours required (in earlier process): | |
| *Total number of issues per* KLOC | = 2.67 × 45 |
| *Total time spent per issue* | = 22 *man-hours* |
| Total time spent on all issues | = 2.67 × 45 × 22 man-hours |
| Man-hours for issues on 22 KLOC of code | = 2643.3 man-hours |
| Round off = 2643 man-hours | |
| Man-hours required (in improved process): | |
| *Total number of issues per KLOC* | = 1.86 × 45 |
| *Total time spent per issue* | = 12 man-hours |
| Total time spent on all issues | = 1.86 × 45 × 12 man-hours |
| | = 1004.4 |
| Delta Man-hours | = 2643 − 1004 |
| | = 1639 |
| Total hours spent by the team for this process improvement | = 240 man-hours |
| Total time saved as a result of this process Improvement | = 1639 − 240 |
| | = 1399 man-hours |

**Table 6** Control mechanism for new process

| S. no. | Description of process inputs | Method\tracking | Frequency | Recommended action | Owner |
|---|---|---|---|---|---|
| 1. | Developer integration testing | Proper template, Schedule tracking if template is being followed and test cases filled completely | Weekly | Tracking and review of template and estimate | Team lead/team member |
| 2. | Prepare test case document | Proper template, Schedule tracking if template is being followed and test cases filled completely | Weekly | Tracking and review of template and estimate | Test lead/tester |
| 3. | Unit test case coverage | Maintain a repository of all the unit test cases along with the production code in version management tool used for the project | Daily | Usage of automation and defect tracking tools | Test Lead |
| | | | | Strictly follow the test cases | |
| | | | | Optimized test cases | |
| 4. | Availability of Hardware component | Maintain a check list in central repository, Remainder mails | Daily update, weekly review | Verification of the received components and update the corresponding check list | Project lead/team lead/team member |
| | | | | Discussion with team to update hardware requirement check list | |
| 5. | Improper fixes/workaround | Defect tracking tool which gives the status of each defect from open to verify and closure | Daily | Proper debugging | Team member |
| | | | | Knowledge sharing from Expert/peer member | |
| | | | | Proper code review from module expert/team Lead | |
| | | | | Trace and update all Review comments | |
| 6. | Legacy issues | Legacy issue repository | Weekly | Knowledge sharing from expert/peer member | Team lead/team member |

**Table 6** Countinued...

| Sl.No. | Description of process inputs | Method\Tracking | Frequency | Recommended action | Owner |
|---|---|---|---|---|---|
| | | | | Discussion with expert to update the issue and track the issues | |
| 7. | Improving debugging process | Adding configurable debug message in the code | Weekly | Classify the debug message modulewise and enable/disable as per the requirements | Team member/team lead |
| | | Identifying the different debugging methods | | Assign severity level to debug message and enable/disable as per the requirements | |
| 8. | Reduce the number of defects by improving the domain/technical competencies of members | By ensuring that the resource competency should meet the project requirements | Domain/ software competencies—quarterly update | Find Training opportunities in the field of domain as well as Software | Team lead/team member |
| | | | Weekly knowledge sharing sessions | Knowledge sharing from peer members | |

### 4.6 Controlling the New Process improvement

In Table 6, we have listed some of the methods to track the process at each stage, recommended action to be taken, when defects or issues occur and finally the responsibility of the person. These steps will help to control and maintain the process.

## 5 Test of Hypotheses

Hypothesis testing helps to make a decision on the basis of sample data and to find whether a hypothesis about the population is true or false. Statisticians have developed several tests of hypotheses (also known as tests of significance) for the function of testing of hypotheses which can be classified as (a) parametric tests or standard tests of hypotheses and (b) nonparametric tests or distribution-free test of hypotheses. Parametric tests usually assume certain properties of the parent/universal population from which we draw samples. So, we have chosen parametric tests for testing our project samples [39].

### 5.1 Important Parametric Tests

The significant parametric tests are (1) $z$ test, (2) $t$ test, (3) $X^2$ test, and (4) $F$ test. All these tests are based on the assumption of normality; that is, the source of data is considered to be normally distributed. Our project sample data size is small; so, we have chosen the $t$ test. The relevant test statistic, $t$, is calculated from the sample data and then compared with its probable value based on $t$ distribution (to be read from the table that gives probable values of $t$ for different levels of significance for different degrees of freedom) at a specified level of significance concerning the degrees of freedom for accepting or rejecting the null hypothesis [39].

*Application Name: Software upgrade scanner copier machine*

**Null Hypothesis** ($\mathbf{H_0}$): There is no significant difference between RPN1 and RPN2, that is, there is no significant difference between current status and revised status.

**Alternate Hypothesis** ($\mathbf{H_1}$): There is significant difference between RPN1 and RPN2, that is, there is significant difference between current status and revised status.

Number of samples $N = 10$, SQRT($N$) = 3.162

A Paired $t$ test was conducted for the following Table. 8, which was taken from the Table 7.

* The $t$ test has been carried out in order to check if there is any significant difference between RPN1 and RPN2 in Table 9.

$P$ value Calculation $t$ value = 2.50 Degrees of Freedom DF = 09

**Table 7** RPN1 TO RPN2 FMEA validation

| Mode of failure | RPN1 | $(x-\mu)$ | $(x-\mu)^2$ | RPN2 | $(x-\mu)$ | $(x-\mu)^2$ |
|---|---|---|---|---|---|---|
| Prepare test case document | 168 | 92.7 | 8593.29 | 84 | 57.9 | 3352.41 |
| Developer integration testing | 120 | 44.7 | 1998.09 | 60 | 33.9 | 1149.21 |
| Improper fixes/workaround | 140 | 64.7 | 4186.09 | 14 | −12.1 | 146.41 |
| Reproduce the issue | 105 | 29.7 | 882.09 | 60 | 33.9 | 1149.21 |
| Availability of hardware component | 60 | −15.3 | 234.09 | 5 | −21.1 | 445.21 |
| Assigning the defect to owner | 56 | −19.3 | 372.49 | 28 | 1.9 | 3.61 |
| Requirement analysis | 40 | −35.3 | 1246.09 | 7 | −19.1 | 364.81 |
| Legacy issues | 25 | −50.3 | 2530.09 | 1 | −25.1 | 630.01 |
| Unit test | 21 | −54.3 | 2948.49 | 1 | −25.1 | 630.01 |
| Missing caveats | 18 | −57.3 | 3283.29 | 1 | −25.1 | 630.01 |
| Total | 753 | | 26274.1 | 261 | | 8500.9 |
| Mean ($\mu$) | 75.3 | | 2627.41 | 26.1 | | 850.09 |
| Standard deviation ($\sigma$) | 51.25827 | | | 29.1563 | | |
| Variance ($\sigma^2$) | 2627.41 | | | 850.09 | | |
| Standard error mean ($\sigma$/SQRT(N)) | 16.21071 | | | 9.220842 | | |

**Table 8** Paired sample statistics for software upgrade scanner copier machine

| Description | | Mean | $N$ | SD | Variance | Standard error mean |
|---|---|---|---|---|---|---|
| Software upgrade Scanner Copier | RPN1 | 75.3 | 10 | 51.2582 | 2627.41 | 16.2107 |
| | RPN2 | 26.1 | 10 | 29.1563 | 850.09 | 9.2208 |

**Table 9** Paired sample test ($t$ test) for software upgrade scanner copier machine

| | |
|---|---|
| Paired difference mean | 49.2 |
| Paired difference standard deviation (s) | 22.10196 |
| Paired difference standard error mean [s/SQRT($N$)] | 6.989869 |
| $s^2/(N-1)$ | 291.9344 |
| Total $\sigma^2/(N-1)$ | 386.3889 |
| SQRT ($s^2/(N-1)$) | 19.65678 |
| $t$ = paired difference mean/SQRT ($s^2/(N-1)$) | 2.502954 |
| Degrees of freedom ($df$) | 9 |
| $t$ table value | 1.833 |

1.83($t$ tabulated Value) < 2.50($t$ Calculated Value), so null hypothesis(H0)is rejected

The two-tailed $P$ value equals $0.0339 < 0.05$, so null hypothesis is rejected.

By conventional criteria, this difference is considered to be statistically significant.

## 5.2 FMEA Validation Result for Software Upgrade Scanner Copier Machine

(i) 1.83($t$ tabulated value) <2.50($t$ calculated value), so null hypothesis is rejected.

(ii) $P$value $= 0.0339 < 0.05$, so null hypothesis is rejected.

(iii) There is significant difference between the process obtained during KLSS implementation and Process improvement after KLSS implementation Shown in Tables 8 and 10.

**Table 10** Confidence limits reports for software upgrade scanner copier machine

| To find confidence limits (2 tailed) | |
|---|---|
| $t$ (0.025,10) | 2.18 |
| Lower level confidence limit | 33.96208267 |
| Upper level confidence limit | 64.43791733 |

(iv) There is higher significance between RPN1 and RPN2, that is, there is difference between current status and revised status.

(v) The results can be used further for continuing improvement process until the process reaches to a satisfactory level.

## 6 Conclusion

In today's highly competitive world, it is indispensable to deliver the software projects in time with optimum cost and good quality. This forces all the software development companies to apply standard quality model and improvement techniques. This paper has discussed the development of a new model, namely the KLSS, which is used to improve the software development process. The framework proposed shows that an application which uses the KLSS model will serve as a better model for finding actual requirements and expectations and the development process has been effectively managed. To compete in service businesses, KLSS will serve as a better model for managing the number of defects or issues, eliminating non-value-added process and producing high-quality software in the software industry. From the literature review, it was found that very limited work had been done on industrial practices, especially software industry for process improvement. This study has addressed the all effective ways of improving the software development process in all aspects. The attempt in the software development process of these models is still in its primitive stage. Some researchers have attempted to modify the existing models for software projects. But they lack practical applicability. The KLSS model presented in this paper has been demonstrated using embedded software application project, and it has been successfully validated using the statistical inference. As regards the future direction, it is suggested that the KLSS quality improvement process model can be implemented in small- and medium-size enterprises (SME), because the results presented in this paper are mainly related to large companies. It will be very useful to assess how SMEs adopt the KLSS models proposed and how these models impact the companies' performance.



## References

1. Sambamurthy, V.; Bharadwaj, A.; Grover, V.: Shaping agility through digital options: reconceptualizing the role of information technology in contemporary firms. MIS Q. **27**(2), 237–263 (2003)
2. Lyytinen, K.; Robey, D.: Learning failure in information systems development. Inf. Syst. J. **9**(2), 85–101 (1999)
3. Lindstroma, L.; Jeffriesb, R.: Extreme programming and agile software development methodologies. J. Inf. Syst. Manag. **21**(3), 41–52 (2004)
4. Kano et al.: Attractive quality and must-be quality. J. Jpn. Soc. Qual. Control **14**(2), 39–48 (1984)
5. Herzberg, F. et al.: The Motivation to Work. Wiley, New York (1959)
6. Matzler, K.; Hinterhuber, H.: How to make product development projects more successful by integrating Kano's model into quality function deployment. Technovation **18**(1), 25–38 (1998)
7. Lee, Y.C. et al.: Kano's model and decision making trial and evaluation laboratory applied to order-winners and qualifiers improvement: a study of computer industry. Inf. Technol. J. **7**(5), 702–714 (2008)
8. Lee, Y.C.; Huang, S.Y.: A new fuzzy concept approach for Kano's model. Expert Syst. Appl. **36**, 4479–4484 (2009)
9. Matzler, K. et al.: How to Delight Your Customers. J. Prod. Brand Manag **5**(2), 6–18 (1996)
10. Eskildsen, J.K.; Kristensen, K.: Enhancing importance-performance Analysis. Int. J. Prod. Perform. Manag. **55**(1/2), 40–60 (2006)
11. Xu, Q. et al.: An analytical Kano model for customer needs analysis. Design Stud. **30**(1), 87–110 (2009)
12. Nilsson-Witell, L.; Fundin, A.: Dynamics of service attributes: a test of Kano's theory of attractive quality. Int. J. Serv. Ind. Manag. **16**(2), 152–168 (2005)
13. Bayraktaroğlu, G.; Özgen, Ö.: Integrating the Kano model, AHP and planning matrix QFD application in library services. Lib. Manag. **29**(4/5), 327–351 (2008)
14. Llinares, G.; Page, A.F.: Kano's model in Kansei Engineering to evaluate subjective real estate consumer preferences. Int. J. Ind. Ergon. **41**, 233–246 (2011)
15. Hu, H.Y. et al.: Amend importance-performance analysis method with Kano's model and DEMATEL. J. Appl. Sci. **9**(10), 1833–1846 (2009)
16. Hoerl, R.: One perspective on the future of Six Sigma. Int. J. Six Sigma Compet. Advant. **1**(1), 112–119 (2004)
17. Antony, J. et al.: Lean Sigma, Manufacturing Engineering, pp. 40–42 (2003)
18. Arnheiter, E.D.; Maleyeff, J.: The integration of lean management and Six Sigma. TQM Mag. **17**(1), 5–18 (2005)
19. Ricondo, I.; Viles, E.: Six Sigma and its link to TQM, BPR, Lean and the learning organization. Int. J. Six Sigma Compet. Advant. **1**(3), 323–354 (2005)
20. Koning, H.de et al.: Lean Six Sigma in healthcare. J. Healthc. Qual. **28**(2), 4–11 (2006)
21. Byrne, G. et al.: A Using a Lean Six Sigma approach to drive innovation. Strategy Leadersh. **35**(2), 5–10 (2007)
22. Carleysmith, S.W. et al.: Implementing Lean Sigma in pharmaceutical research and development: a review by practitioners. R&D Manag. **39**(1), 95–106 (2009)
23. Pepper, M.P. et al.: The evolution of Lean Six Sigma. Int. J. Qual. Reliab. Manag. **27**(2), 138–155 (2010)
24. Gamal Aboelmaged, M.: Reconstructing Six Sigma barriers in manufacturing and service organizations. Int. J. Qual. Reliab. Manag. **28**(5), 519–541 (2011)
25. Vinodh, S. et al.: Implementing Lean Sigma in an Indian rotary switches manufacturing organization. Prod. Plan. Control **25**, 288–302 (2014)
26. Vinodh, S.; Prasanna, M.: Lean Six Sigma in SMEs: an exploration through literature review. J. Eng. Design Technol. **11**(3), 224–250 (2013)
27. Assarlind, M. et al.: Multi-faceted views on a Lean Six Sigma application. Int. J. Qual. Reliab. Manag. **30**(4), 387–402 (2013)
28. Xu, Q. et al.: An analytical Kano model for customer need analysis. J. Design Stud. **30**(1), 87–110 (2009)
29. Sharma, U.: Implementing Lean principles with the Six Sigma advantage: how a battery company realized significant improvements. J. Organ. Excell **22**, 43–52 (2003)

30. Furterer, S.; Elshennawy, A.: Implementation of TQM and Lean Six Sigma tools in local government: a framework and a case study. Total Qual. Manag **16**, 1179–1191 (2005)
31. Marti, F.: Lean Six Sigma method in phase 1 clinical trials: a practice example. Qual. Assur. J. **9**, 35–39 (2005)
32. Chao, T.S. et al.: Improving service quality by capitalising on an integrated Lean Six Sigma methodology. Int. Pap. Six Sigma Compet. Advant **2**, 1–22 (2006)
33. Kumar, M. et al.: Implementing the Lean Sigma framework in an Indian SME: a case study. Prod. Plan. Control **17**, 407–423 (2006)
34. D'Angelo, R.; Zarbo, R.J.: The Henry Ford production system: measures of process defects and waste in surgical pathology as a basis for quality improvement initiatives. Am. J. Clin. Pathol. **3**, 423–429 (2007)
35. Tonini, A.C. et al.: An application of Six Sigma with Lean production practices for identifying common causes of software process variability. In: PICMET Proceedings, 5–9 August, Portland, Oregon, PICMET, pp. 2482–2490 (2007)
36. Kanakana, M.G. et al.: Lean Six Sigma framework to improve throughput rate, 978-1-4244-6484-5/10, IEEE (2010)
37. Pillai, A.K.R. et al.: Implementing integrated Lean Six Sigma for software development: A flexibility framework for managing the continuity: change dichotomy. Glob. J. Flex. Syst. Manag. **13**(2), 107–116 (2012)
38. Pillai, A.K.R. et al.: Improving information technology infrastructure library service delivery using an integrated Lean Six Sigma framework: a case study in a software application support scenario. J. Softw. Eng. Appl. **7**, 483–497 (2014)
39. Kothari, C.R.: Research Methodology Methods and Techniques, 2nd edn. New Age International, New Delhi (2004)