ASSIGNMENT 4

4

**DMAIC PLAN / STATUS**

CSE 6329 -- SOFTWARE MEASUREMENT AND QUALITY ENGINEERING

Professor Dennis J. Frailey

**Fall, 2019**

|  |  |
| --- | --- |
| NAME(s) | ID Number(s) |
| **Goutami Padmanabhan** | **1001669338** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Grading Comments (student – do not write inside this box)** | | | |
| **<total goes here>** | 1. **Define -- CTQs (Critical to Quality) (10 points)** | | |
| **/ 5** |  | |
| **/ 5** |  | |
| **<total goes here>** | 1. **Measure – Organizations, Process Flow and Swim Lane Diagram (20 points)** | | |
| **/ 8** | **Table of People and Organizations** |  |
| **/ 12** | **Swim Lane Diagram** |
| **<total goes here>** | 1. **Analyze Response Problem-- Root Cause Analysis (30 points)** | | |
| **/ 10** | **RCA Method 1** |  |
| **/ 10** | **RCA Method 2** |
| **/ 5** | **Table of Causes (including Root Causes)** |
| **/ 5** | **Cause Map** |
| **<total goes here>** | 1. **Analyze Failure Rate Problem – Data Analysis (30 points)** | | |
| **/ 10** | **Analysis Phase 1 – manager, language, method** |  |
| **/ 10** | **Analysis Phase 2 – size, release date** |
| **/ 5** | **Root Cause(s)** |
| **/ 5** | **Analysis Phase 3 - Cause Map** |
| **<total goes here>** | 1. **Improve -- Recommendations (10 points)** | | |
| **/3** | **Most Important Root Causes** |  |
| **/ 3** | **Recommendations** |  |
| **/ 4** | **Legibility, Correct English, etc.** |
| **<total>** | **Grand Total** | | |

**DMAIC Plan / Status**

This document consists of our DMAIC plan and the results obtained so far, so as to provide a status report on execution of that plan.

**DEFINE**

**Charter (from management):**

**Business Problem:** The customers are complaining that there are increasing numbers of failures in our newer products and that correction of software failures is too slow. Our most important customers are the ones complaining the most about this problem, although we are hearing from other customers as well. We must correct this in order to satisfy our customers.

**Goal:** Determine the causes of the slow response and higher failure rate and correct them. Reduce the response time by at least 50% and reduce failure rate to what it has traditionally been in the past (or better).

**Scope:** The entire business process of the company may be affected by this. No part of the company is off limits.

**Timeline:** We must have a complete plan and analysis by May 10.

**Resources:** We have obtained the assistance of several UTA students who have taken a course that covered the appropriate methods. We want them to develop a more complete DMAIC plan.

**Definition (from team assigned to solve the problem)**

**The problems:**

1) Slow response to software failures and

2) higher failure rates.

**The customers:** Several, notably Acme Corporation and Zephyr.

**Voice of the customer:** See memos from Acme and Zephyr. The customer quality requirement is software that runs properly, with minimal failures, and that there is rapid response to correcting the software when it does fail.

**CTQs:** Measurable attributes that are critical to quality for our customers.

| CTQ #1 (slow response) | How Measured | Why it is Critical |
| --- | --- | --- |
| **The Time required to correct or rectify the faults in software which is resulting in failures and the time required to respond to the notification of failure by the client.** | **This CTQ can be measured by counting the number of days starting from the day when the customer reports a problem to the organization, to the day on which the representative of the company sends the version of the software which is fault free and gets installed in the system located at the customer premises.**  **Formula:**  **Time Required to Correct Fault (days) = (Problem Reporting Day – Day on which Fault Free Software Delivered and Installed)** | Time required to correct error and install error free software is very critical to the client/customer as their business depends on the software which is delivered to them and delay in correcting those faults can cause delay and directly impact their daily business operations resulting in financial loss. |

| CTQ # 2 (high failure rate) | How Measured | Why it is Critical |
| --- | --- | --- |
| **The Quality of the Product or Product Quality.** | Quality of the product can be measured directly from the number of failures reported for a particular software product.  Formula:  Product Quality = No of Failures or Defects reported by the client or customer to the organization | **Product Quality is one of the critical factors which contributes to the success of an organization in terms of continuous business operations and poor or subpar Product quality can result in reducing the number of customers and can eventually affect the reputation and financial results of the company (IPC).**  Also, if the software product is delivered with high quality in the initial phase of delivery itself, then it can save us a lot of time which we would spend in its maintenance and correction of defects/failures. As a result, improve customer satisfaction and the software team of IPC can focus more on improving the quality of the product and meet all demands and requests from the customer. |

**Target process(es) to be improved:** All processes in the company. Note: we will begin by analyzing the customer support process but any part of our company’s process may be subject to improvement based on findings from our analysis. Specific improvements are defined as a result of the analysis.

**Project Targets:** Reduce response time by at least 50%; reduce failure rate to at least the historical average (preferably better).

**MEASURE processes and products.**

**2.1 Measure the customer support process (response time problem)**

**2.1.a Organizations/People (Roles) Interviewed:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 2.1.a – Table of People and Organizations** | | | |
| **Name** | **Organization** | **Job Title / Role** | **Summary of what they do** |
| Lars Johansson | ACME Corporation | Information Technology Manager | Responsible for providing computing and networking services to all ACME employees |
| Peter Patel | Zephyr Corporation | Software Acquisition Manager | Responsible for the management of the company’s software and functioning. |
| James Johnson | International Products Corporation (IPC) | Customer Representative in Marketing and Sales Division | IPC’s representative to some of its major customers, including ACME, Delta, Zephyr and Zeta Corporation |
| George Wilson | International Products Corporation (IPC) | Maintenance Clerk in Maintenance and Support Division | Review incoming problem reports, prioritize them, combine similar reports that probably require the same fix, and ship updates back to the customer rep. |
| Melinda Shah | International Products Corporation (IPC) | Division Manager at Maintenance and Support Division | Resolves priority disputes with problem reports and to assure that there are adequate people and resources for testing of all repairs. And adequate programming resources to handle the level of problems being reported. She also sits on CCB and approves all the proposed repairs. |
| Sharleen Jefferson | International Products Corporation (IPC) | Maintenance Programming Manager at Maintenance and Support Division | Supervises the programmers and assigns them to work on specific problem report repairs. Also sits on the CCB. |
| Narayan Bhat | International Products Corporation (IPC) | Maintenance Programmer at Maintenance and Support Division | Works for Sharleen, the maintenance programming manager. Analyzes problem reports, identify the source of the problem, and make the required repairs. |
| Wendy Stottlemeyer | International Products Corporation (IPC) | Maintenance Test Team Manager at Maintenance and Support Division | Reports to Melinda (maintenance manager) performs system level tests on all repairs to see if they work, to run regression tests, and to prepare final update packages for completed and approved repairs. |
| Jeff Arterburn | International Products Corporation (IPC) | Division Manager  At Quality Assurance Division | Supervises QA staff and serves on both software development CCB and software maintenance CCB |
| Rachel Wallace | International Products Corporation (IPC) | Division Manager  At Software Development Division. | Supervises all software development for new products |
| David Kanappell | International Products Corporation (IPC) | Manager of Financial Product Software Development  at Software Development Division. | Manages the product development team for financial products. |
| Leticia Benavides | International Products Corporation (IPC) | Manager of Payroll Products Software Development  at Software Development Division | Manages the product development team for payroll products. |
| James Donohu | International Products Corporation (IPC) | President and CEO at Office of the President Division | Chief Executive Officer of IPC. |

**2.1.b Process Flow:**

S**tep 1:** Whenever there is failure attributed to the software provided by the IPC, Lara Johansson, the Information Technology Manager at ACME Corporation sends the particulars i.e. the reports about failures to the customer representative, Mr. James Johnson at IPC.

**Step 2:** James Johnson, the customer representative collects the relevant information about the software failure and completes a problem report (online). James Johnson also assigns a severity level to that problem.

**Step 3:** The report made online by James Johnson then goes to the maintenance clerk George Wilson.

**Step 4:** If the problem reported is severe and classified as critical halting the customers business operations, James Johnson contacts directly Melinda Shah who is the Maintenance Manager and she in turn sends a Software Developer to the customer site to fix the problem.

**Step 5:** James Johnson can see the status of the problem report online and give the weekly status reports or immediate report to the customer (when requested).

**Step 6:** George Wilson reviews the incoming problem reports sent by James Johnson, assigns a unique tracking number to each problem and gives each problem a priority based on their severity on a scale of minor, normal and critical.

**Step 7:** George contacts Maintenance Manager Melinda Shah when he doesn’t agree with the severity code assigned by James Johnson. This sometimes takes a week **time** because Melinda has to discuss this with James Johnson and both of their schedules don’t match always.

**Step 8:** If the problem is critical then a person is sent to the customer site by Sharleen, the maintenance programming manager when contacted by Melinda. For the other two severity levels categories they are put in the queue for that day with that priority.

**Step 9:** Initially the problem will be in “waiting to be assigned stage” and **t**hen the problem is moved to “Assigned and Waiting to be Fixed Stage” when a programmer is assigned to it. If George finds that a problem is similar to the problem reported earlier, he links the new problem to the previous reported problem.

**Step 10:** George will unlink the new problem, If the linked problem is reported to be not linked by Maintenance programmer and it becomes a standalone problem which gets added to the queue on that day.

**Step 11:** Then the problem moves to fixed stage once the repair is made and waits for regression test and then moves to tested and waits for final approval. If not approved, it goes back to waiting queue and assigned to same programmer with top priority.

**Step 12:** Melinda Shah sits on the configuration control board (CCB), who approves all proposed repairs (repair plan approval, preliminary repair approval and final repair approval).

**Step 13:** Melinda Shah resolves any disputes regarding priority of a problem repair and discusses this with Sharleen and James Johnson.

**Step 14**: Sharleen assigns the problems to the maintenance programmers which are needed to be fixed especially the products which have high failure rates.

**Step 15:** Narayan Bhatt, the maintenance programmer analyzes the problem assigned by Sharleen and look at the problem report and tries to reproduce the problem.

**Step 16:** Narayan Bhatt debugs the software till he figures out what’s wrong and sometimes in order to understand needs design and requirements, consults with the original programmer.

**Step 17:** Narayan Bhatt then proposes the solution to the CCB for approval.

**Step 18:** CCB checks whether the proposed solution causes any other problems and If they find any problem, then Narayan Bhatt will try a different solution and

present it to them in the next meeting.

**Step 19:** Narayan implements the solution once approved by CCB and comes up with new unit test cases and performs the unit test results and solution to the CCB for approval.

**Step 20:** Narayan Bhat gives the Software to the test group to perform the system tests and regression tests. Then now he starts working on another problem.

**Step 21:** If the test team finds a problem, then they put it back in the queue with highest priority so that is assigned to the programmer once done with the current problem it’s the next thing he is assigned to do.

**Step 22:** Once the software has been repaired, the testing team performs the system level tests and then unit tested by the programmer. Regression tests are carried out only when all current repairs on that software has been completed.

**Step 23:** If there are no problems during regression tests, CCB gives the final approval.

**Step 24:** The package which is repaired is then sent to George Wilson.

**Step 25:** Sharleen prepares the updated installation package which usually takes a day or two and then changes the status of the problem report to “waiting for shipment”.

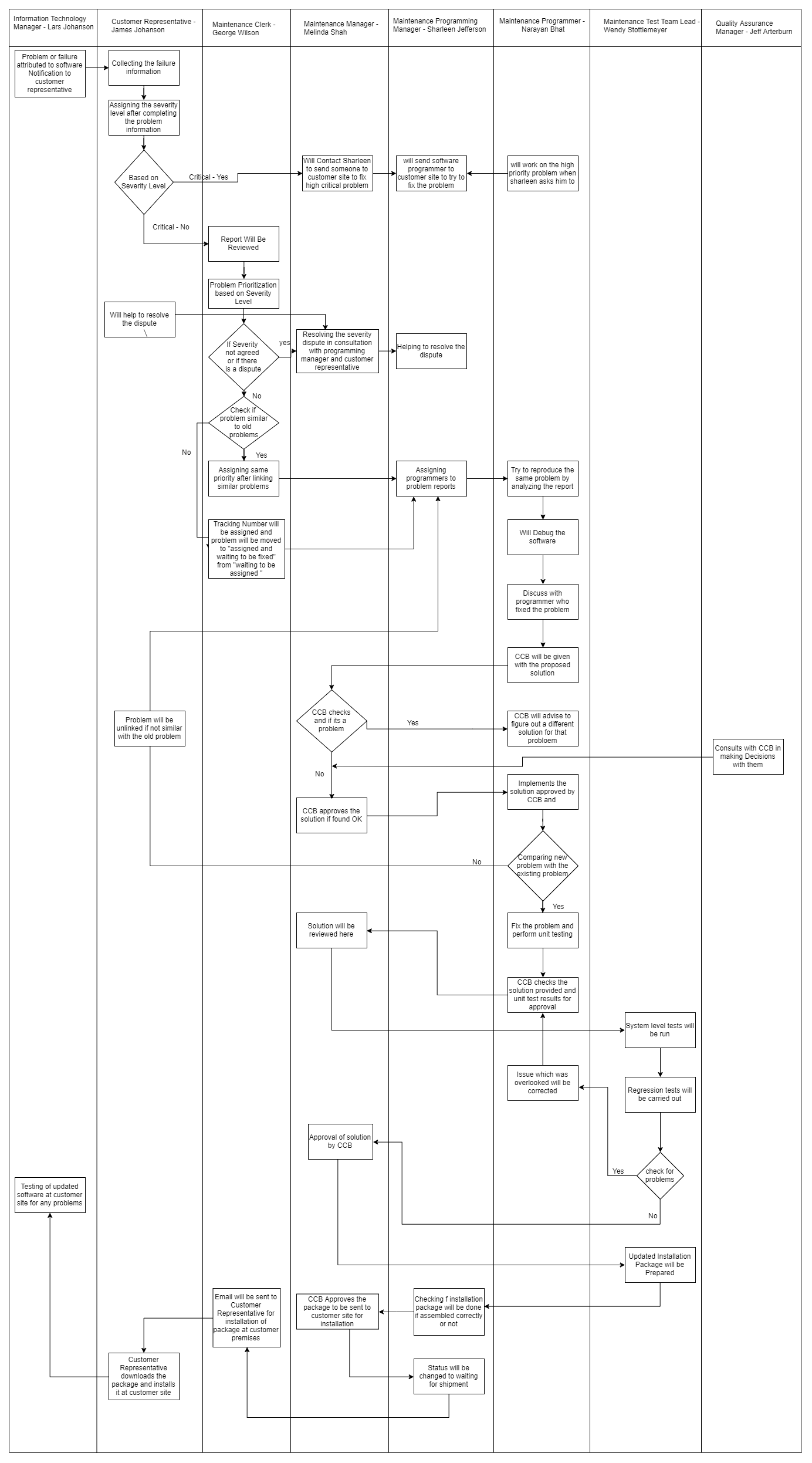
**Step 26:** Melinda shah and Sharleen Jefferson checks if the installation package has been assembled correctly and CCB gives the final approval for sending the update back to the customer.

**Step 27:** An email is sent to the James Johnson, the customer representative who downloads the installation package.

**Step 28:** James Johnson installs the downloaded installation package on the customer’s site and when the permanent repair is delivered by the team, he installs it on the customer’s site.

**Step 29:** If there is any problem with the repair, it is sent back to the home office, so they can see or analyze what went wrong and work on it to fix the problem.

**Step 30:** The client/customer performs tests on the software installed by James Johnson, the customer representative and check if the reported problem is rectified or not and ensure everything works smoothly so as to have continuous business operations without interruptions.

**Swimlane Diagram:**

**2.1.c: Data to Collect:**

* Interviews with key people involved in the process described above. Interviews are intended to extract their perspectives on the process as well as numeric data, where available.
* International Products Corporation Data relevant to this problem.

Interview results have been collected and are summarized in various supplementary files. Product data are found in the **IPC Data** and **IPC Defect Data** Reports.

**ANALYZE**

**3 Analyze Slow Response Problem:**

**3.1 Root Cause Analysis:**

Root Cause Analysis (RCA) is defined as a business process to find out where it went wrong. The root cause analysis has various techniques and some of them are discussed here as follows:

* Five Whys: It refers to asking “Why? questions” till we find the root cause for a particular problem or a set of problems.
* Cause and Effect Charts: It involves categorizing the causes and then subdivide them to get a better and clear understanding of what causes are the ones that really cause the problem and have an effect.
* Fault Tree Analysis: It is a Top down, deductive failure analysis in which an undesired state of a system is analyzed using Boolean logic in order to combine a series of lower level events.
* Matrix Diagram: It is a tool which allows a team to identify the presence and strengths of relationships between two or more lists of items by providing a compact way of representing many to many relationships.

We have selected **Five Whys** and **Fault Tree Analysis** techniques from the above listed RCA Techniques in order to find out the root causes of the customer’s problem.

**1) FIVE Whys Technique:**

* It is a technique used to determine the root cause of the problem. It is done by constantly asking the question in the form of ‘Why?’.
* The question which is being asked forms the basis for the next question and is an interrogative iterative process. It is used to explore the cause and effect relationship for the underlying specific problems.
* Some potential cases and problems which we have identified are listed below.

Causes which were found using Five Why’s Technique:

Problem 1: Slow response in Providing Solutions

**Why: Company’s problem report tracking system (Root cause)**

Why: Some problems take too long to fix.

Why: Linking of problems gets delayed as linking is done without consulting maintenance programmers.

Why: Unlinking problem sends it back in Queue.

Why: There is no information about the original data.

Problem 2: Slow Response in validation of repairs

**Why: No proper infrastructure to save test cases/code. (Root cause)**

Why: Too much time is taken to validate the repairs.

Why: Delay because of testers.

Why: Require new set of unit and regression test cases.

Why: Old test cases are not saved or maintained in order to expedite the testing process.

Problem 3: Delay in providing the solution

**Why: Company is following old policy in producing the solution. (Root Cause)**

Why: Delay in producing the solution or providing the solution.

Why: Final approval from CCB involved higher wait time.

Why: Approval to be done which requires the signature of all members on the same day.

Why: CCB members are not available at the same time.

Problem 4: Delay in assigning tasks

**Why: Company’s Old Policy. (Root Cause)**

Why: Assigning tasks gets delayed

Why: Problems with assigning programmer

Why: The most suitable programmer is not available as the good ones are busy with other priority tasks and cannot take up the new ones until they finish the already working one.

Why: Company is not allowing multi-tasking.

Problem5: Difficulty in solving the problem faced by maintenance programmers

**Why: Higher priority is given to rapid software development Process rather than maintenance by the company (Root cause).**

Why: Assumptions are made by maintenance programmers which might be wrong and may require rework too.

Why: There is a lack of documentation.

Why: Maintenance programmers have to Consult Original Programmer who wrote the code but Original programmers are not in a position to understand his or her own code.

Why: Lack of documentation means company has switched their development approach to “Agile Methodology or Technique”.

**2. Fault Tree Analysis**

Fault Tree Analysis is mainly used to represent all possible causes in one diagram. For this we need to,

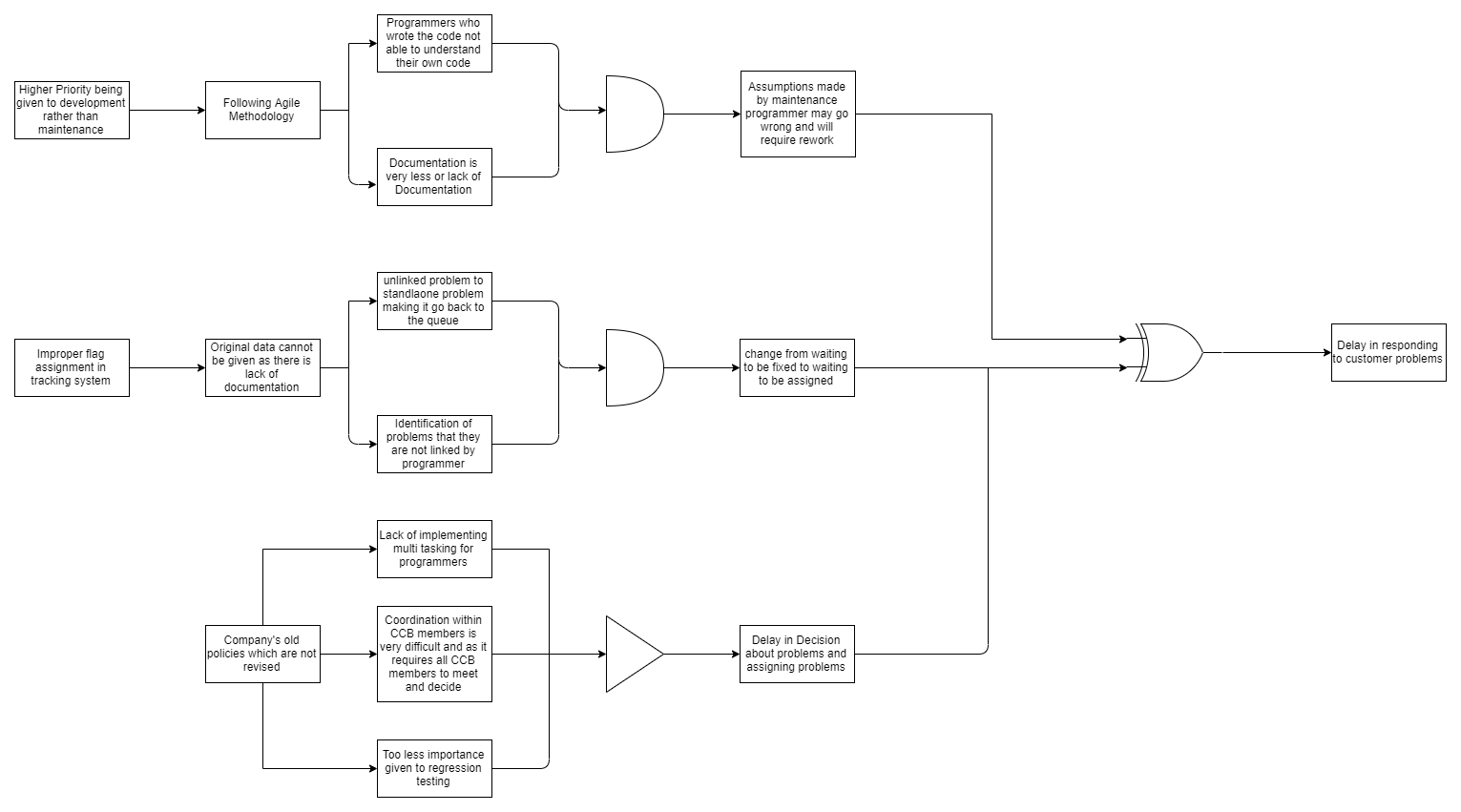
1. Define the fault condition and write down the top-level failure.
2. Determine the possible reasons for the occurrence of the failure.
3. Continue to break down each element along with additional gates to lower levels and also consider the relationships between the elements in order to help you decide whether to use an "and" or an "or" logic gate.

Analysis diagram has three types of symbols.

* 1. The first “and” symbol is where all the options are causes.
  2. The second “OR” symbol is where the options might be a cause.
  3. The last symbol “Exclusive OR” is where exactly one of the options could be the cause.

Based on our understanding we have clearly depicted the same in the form of a fault tree analysis diagram. The ultimate problem and the one which is reportedly said to be occurring again and again is that the customer complaints about “delayed response to customer reports of failure”.

The causes that are responsible for this problem are lack of documentation, source code documentation and saved unit test cases. More of the management issues like delay in approvals by CCB, company not allowing multitasking, and regression tests being performed less often.



From the above diagram, the root causes that are identified are,

1) Keeping development on a higher priority over maintenance issues and focusing more on developing more new products.

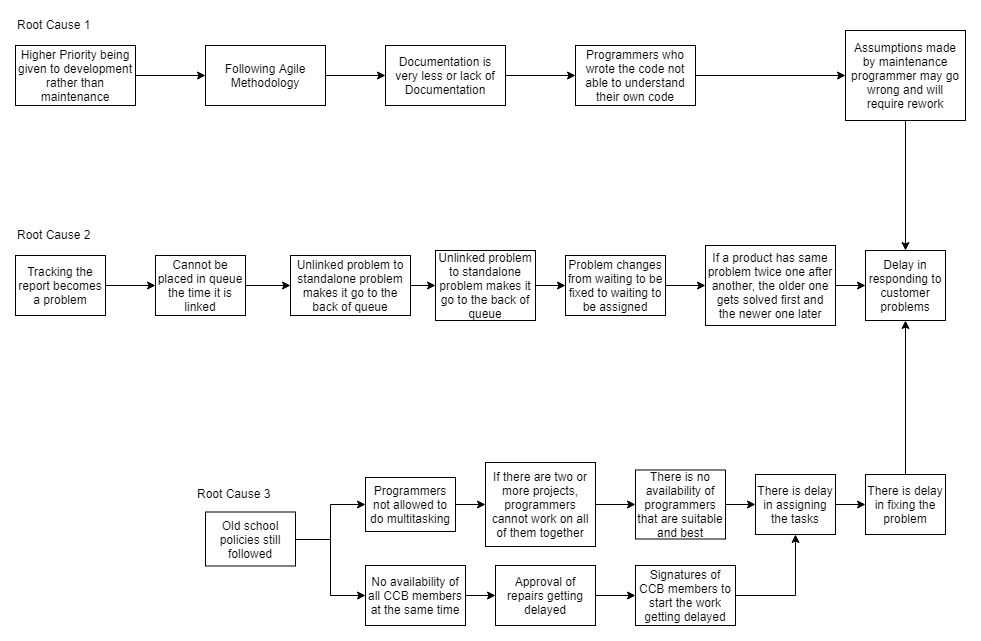
2) Improper and Inefficient functioning of tracking system especially the flagging system.

3) The company following old policy of not allowing multi-tasking, late approvals by CCBs and not too many regression test being performed.

| Table 3.3 – Summary of Causes (# = root cause) | | | |
| --- | --- | --- | --- |
| Cause  (# indicates a root cause) | Description | Importance (High, Medium, Low) | Reasons for indicated level of Importance |
| 1. | Higher Priority given to development rather than maintenance | medium | Fixing of Older products should be focused Instead of focusing on the development of newer products. Else it may result in financial loss to the customer. Also, it will affect the reputation of the company and client. |
| 2. | Tracking the reported problem becomes a problem | High | This is given High priority because it is The major cause of why response to customer complaints gets delayed is because the problems are not solved within the time frame due to inappropriate tracking/flagging of problems by the system based on severity level. |
| 3 | The company follows old school policies | Low | The company policies have to be amended to allow the programmers to work on more than one project and CCB approval policies has to be modified to make the approvals and also to make the installation of fix for the defects faster. However over adhering to company policies can lead to problems which can contribute to slow response time. |

3.4 Cause Map:

A causal map is used when there are multiple root causes or complex situations.



**3.5 Collect Data:** To be performed, if necessary, to identify most critical root causes.

IPC Data consists the information regarding the response time, but there is no measure for the time taken to fix the problem. Data needs to be collected which records the fixing time and time taken at every action, this helps us to analyze the problem in much better way. There is also no measure to find if there is communication with the customer at regular intervals. We can collect data on how god the systems and tools used at IPC, which may be a causing factor for the delay in response.

**4 Analyze High Defect Level Problem:**

**4.1 Data Analysis Phase 1: Analyze Manager, Method and Language** to see if any of these relates to the defect level.

Let’s discuss some of methods of analysis used The data has been analyzed in these different types

1. Scatter Chart and Bar Graph for Product Quality Analysis Yearly.

2. Bar Graph for Product quality in terms of programming Language used.

3. Product quality in terms of development method used.

4. Sorting the data according to the number of team members or team size

5. Sorting the data according to the manager who handled the project.

6. Sorting the data showing only the products which showed high defects and which the customers complained about.

We have performed the analysis using Scatter plots and Bar graphs to analyze which factors are ultimately leading to high defects rate.

| Table 4.1 – Summary of Phase 1 Data Analysis  Identifying Factors Related to High Defect Rate Problem | | |
| --- | --- | --- |
| Data Analysis Method | Description (what data were used for each analysis, what refinement, calculations, and analysis methods were used) | Conclusions reached from this analysis method (including methods that resulted in no conclusion) |
| Scatter plot & Bar Graphs | Product Quality Analysis Yearly.  For the scatter plot, we show the critical defects/KLOC, escaping defects/KLOC, normal defects/KLOC, minor defects/KLOC for each month and analyzed.  Also we calculated the average critical defects/KLOC, average escaped defects/KLOC, average normal defects/KLOC, average minor defects/KLOC for each of the years 2015,2016,2017 and plotted the bar chart. | The Escaping defects were at the maximum between 1st Jan 2015 to 1st March 2015 and it is also highest among all defects every month as it is the total of critical, minor and normal defects. |
| Bar Graphs | Product quality in terms of programming Language used.  The average escaped defects/KLOC, critical defects/KLOC, normal defects/KLOC, minor defects/KLOC versus the programming languages used i.e. Python and Ruby was plotted and the bar graph is drawn. | The programming language used is not the main root cause for having high defects. The escaping defects though are higher than the other defects and higher when using Python than Ruby in the year 2015. |
| Bar Graphs | Product quality in terms of development method used.  The average escaping defects/KLOC, critical defects/KLOC, normal defects/KLOC and minor defects/KLOC against the development methods used i.e. SCRUM and Extreme Programming was plotted on the bar graph. | The defects/KLOC were on the higher side in case of Extreme Programming than SCRUM and thus Development Method is one of the main reason for high defect rate. |
| Sorting Data | Sorting the data according to the number of team members or team size | The number of people in a team or the size of the team does not affect the number of escaping defects. |
| Sorting Data | Sorting the data according to the manager who handled the project. | The products handled by Manager Gomez had a Total number of escaping defects lesser than the products handled by Manager Kanappell and thus Manager handling the projects had an impact on the number of escaping defects or the product quality. |
| Sorting data | Sorting the data showing only the products which showed high defects and which the customers complained about. | The number of escaping defects for the products which the customers complained about were on the higher side. Out of these, most of the products were developed using SCRUM and only 2 were done by following Extreme Programming. Thus, the development process had its impact on product quality which resulted in receiving customer complaints. |

**1.** **Scatter Chart and Bar Graph for Product Quality Analysis Yearly**

In this method, we took the actual data i.e., for the analysis, and plotted the below graph showing the critical defects/KLOC, escaping defects/KLOC, normal defects/KLOC, minor defects/KLOC for each month and analyzed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rel Date | Escaping Defects / KLOC | Critical Defects / KLOC | Normal Defects / KLOC | Minor Defects / KLOC |
| 1-Jan-15 | 1.643 | 0.357 | 0.857 | 0.429 |
| 1-Feb-15 | 2.416 | 0.449 | 1.180 | 0.787 |
| 1-Mar-15 | 1.546 | 0.305 | 0.762 | 0.479 |
| 1-May-15 | 1.518 | 0.325 | 0.759 | 0.434 |
| 1-Jun-15 | 1.386 | 0.273 | 0.682 | 0.432 |
| 1-Jul-15 | 1.191 | 0.225 | 0.607 | 0.360 |
| 1-Aug-15 | 1.259 | 0.223 | 0.629 | 0.406 |
| 1-Oct-15 | 0.949 | 0.194 | 0.475 | 0.281 |
| 1-Nov-15 | 0.933 | 0.193 | 0.461 | 0.279 |
| 1-Dec-15 | 1.995 | 0.420 | 0.998 | 0.578 |
| 1-Jan-16 | 1.103 | 0.207 | 0.552 | 0.345 |
| 15-Feb-16 | 1.263 | 0.184 | 0.684 | 0.395 |
| 1-Mar-16 | 1.012 | 0.194 | 0.499 | 0.319 |
| 1-May-16 | 2.143 | 0.333 | 1.143 | 0.667 |
| 15-Jun-16 | 1.804 | 0.333 | 0.882 | 0.588 |
| 1-Jul-16 | 0.863 | 0.169 | 0.469 | 0.225 |
| 1-Aug-16 | 1.027 | 0.202 | 0.495 | 0.330 |
| 1-Oct-16 | 1.000 | 0.210 | 0.484 | 0.306 |
| 1-Nov-16 | 1.028 | 0.211 | 0.479 | 0.338 |
| 1-Dec-16 | 1.166 | 0.190 | 0.571 | 0.404 |
| 1-Jan-17 | 0.938 | 0.188 | 0.469 | 0.281 |
| 28-Feb-17 | 0.896 | 0.179 | 0.458 | 0.259 |
| 1-Mar-17 | 1.691 | 0.345 | 0.836 | 0.509 |
| 1-May-17 | 1.756 | 0.333 | 0.889 | 0.533 |
| 1-Jun-17 | 0.982 | 0.196 | 0.464 | 0.321 |
| 30-Jul-17 | 0.804 | 0.143 | 0.393 | 0.268 |
| 1-Aug-17 | 0.951 | 0.230 | 0.475 | 0.246 |
| 1-Oct-17 | 1.911 | 0.370 | 0.963 | 0.578 |
| 1-Nov-17 | 1.386 | 0.280 | 0.680 | 0.426 |
| 1-Dec-17 | 1.095 | 0.223 | 0.527 | 0.345 |

From the graph we can see that the escaping defects were at the maximum between 1st Jan 2015 to 1st March 2015 and it is also highest among all defects every month as it is the total of critical, minor and normal defects. Also we calculated the average critical defects/KLOC, average escaped defects/KLOC, average normal defects/KLOC, average minor defects/KLOC for each of the years 2015,2016,2017 and plotted the bar chart as shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Avg Values/Year** | **Escaping Defects / KLOC** | **Critical Defects / KLOC** | **Normal Defects / KLOC** | **Minor Defects / KLOC** |
| 2015 | 1.48 | 0.30 | 0.74 | 0.45 |
| 2016 | 1.24 | 0.22 | 0.63 | 0.39 |
| 2017 | 1.24 | 0.25 | 0.62 | 0.38 |

It can be seen from the above bar chart that, the escaping defects(average value) are the most in all 3 years, with 2015 being the highest in terms of average escaped defects. Secondly, it can also be seen that the normal defects were more in all the 3 three years than critical and Minor defects only.

**2. Bar Graph for Product quality in terms of programming Language used**

In this method of analysis, the defect data was analyzed in terms of the Programming language used.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Python** | | | | **Ruby** | | | |
| **Avg Values/Year** | **Escaping Defects / KLOC** | **Critical Defects / KLOC** | **Normal Defects / KLOC** | **Minor Defects / KLOC** | **Escaping Defects / KLOC** | **Critical Defects / KLOC** | **Normal Defects / KLOC** | **Minor Defects / KLOC** |
| **2015** | 1.54 | 0.31 | 0.77 | 0.46 | 1.24 | 0.25 | 0.61 | 0.38 |
| **2016** | 1.23 | 0.22 | 0.63 | 0.38 | 1.25 | 0.22 | 0.62 | 0.40 |
| **2017** | 1.31 | 0.26 | 0.65 | 0.40 | 1.25 | 0.25 | 0.62 | 0.39 |

The average escaped defects/KLOC, critical defects/KLOC, normal defects/KLOC, minor defects/KLOC versus the programming languages used i.e. Python and Ruby was plotted and the bar graph is shown below.

The Escaping defects/KLOC are on the higher side in case of product developed using Python especially in the year 2015 and slightly higher in 2017 than Ruby. But escaping defects/KLOC is higher when the product is developed with Ruby in the year 2016 than Python. Also, the Normal defects/KLOC, Critical defects/KLOC and Minor defects/KLOC is slightly higher only in the year 2015 when product developed using Python than Ruby and both the languages Python and Ruby have about same number of defects in other years. So, it can be concluded that the programming language used is not the main root cause for having high defects. The escaping defects though are higher than the other defects and higher when using Python than Ruby in the year 2015.

**3**. **Product quality in terms of development method used.**

In this method of analysis, the defect data was analyzed in terms of the development method used.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Extreme** | | | | **Scrum** | | | |
| **Avg Values/Year** | **Escaping Defects / KLOC** | **Critical Defects / KLOC** | **Normal Defects / KLOC** | **Minor Defects / KLOC** | **Escaping Defects / KLOC** | **Critical Defects / KLOC** | **Normal Defects / KLOC** | **Minor Defects / KLOC** |
| **2015** | 1.83 | 0.36 | 0.92 | 0.55 | 1.25 | 0.25 | 0.62 | 0.38 |
| **2016** | 1.24 | 0.22 | 0.63 | 0.39 | 0.00 | 0.00 | 0.00 | 0.00 |
| **2017** | 1.69 | 0.35 | 0.84 | 0.51 | 1.19 | 0.24 | 0.59 | 0.36 |

The average escaping defects/KLOC, critical defects/KLOC, normal defects/KLOC and minor defects/KLOC against the development methods used i.e. SCRUM and Extreme Programming was plotted on the bar graph and it is shown below.

From the bar charts, It can be observed that average escaping defects/KLOC and average normal defects/KLOC is higher in case of Extreme Programming than SCRUM in 2015 and 2017 as SCRUM was not followed in 2016 overall. The average critical defects/KLOC and average minor defects/KLOC was slightly higher in case of Extreme Programming than SCRUM. Thus, we can conclude that defects /KLOC were on the higher side in case of Extreme Programming than SCRUM and thus Development Method is one of the main reason for high defect rate.

4.  **Sorting the data according to the number of team members**

In this method of analysis, we sorted the below data by Team Size (Number of People in a Team) and then analyzed the data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Team Size (people)** | **Product** | **Manager** | **Method** | **Language** | **Size (KLOC)** | **Escaping Defects** |
| **30** | **Halo** | **Gomez** | **SCRUM** | **Python** | **103.2** | **98** |
| 24 | Inspect | Gomez | SCRUM | Ruby | 93.2 | 87 |
| **22** | **Zorro** | **Gomez** | **SCRUM** | **Ruby** | **75.0** | **104** |
| 20 | Storage | Gomez | Extreme | Python | 71.0 | 73 |
| **20** | **Zillow** | **Kanappell** | **SCRUM** | **Python** | **67.5** | **129** |
| 18 | Mu | Gomez | Extreme | Python | 72.2 | 73 |
| 18 | River | Gomez | Extreme | Ruby | 62.0 | 62 |
| **18** | **Zap** | **Kanappell** | **SCRUM** | **Ruby** | **61.0** | **58** |
| 16 | Optimize | Kanappell | Extreme | Ruby | 51.0 | 92 |
| **16** | **Quorum** | **Gomez** | **Extreme** | **Ruby** | **54.6** | **56** |
| **16** | **Window** | **Kanappell** | **Extreme** | **Ruby** | **55.0** | **93** |
| 16 | Yank | Gomez | SCRUM | Ruby | 56.0 | 55 |
| **14** | **Precise** | **Gomez** | **Extreme** | **Python** | **53.3** | **46** |
| 14 | Value | Gomez | SCRUM | Ruby | 50.2 | 45 |
| **14** | **Xray** | **Kanappell** | **SCRUM** | **Ruby** | **45.0** | **79** |
| 14 | Zebra | Gomez | SCRUM | Ruby | 56.0 | 45 |
| 14 | Zunder | Kanappell | SCRUM | Python | 49.3 | 54 |
| **12** | **Casper** | **Kanappell** | **SCRUM** | **Ruby** | **45.9** | **71** |
| 12 | Epsilon | Kanappell | SCRUM | Python | 44.0 | 61 |
| 12 | Framer | Kanappell | SCRUM | Python | 44.5 | 53 |
| 12 | Guide | Gomez | Extreme | Python | 49.3 | 62 |
| 12 | Justify | Kanappell | Extreme | Ruby | 38.1 | 76 |
| 10 | Delta | Kanappell | SCRUM | Python | 36.9 | 56 |
| 10 | Lambda | Gomez | Extreme | Ruby | 38.0 | 48 |
| 10 | Time | Kanappell | Extreme | Ruby | 42.0 | 49 |
| 8 | Kappa | Gomez | Extreme | Python | 29.0 | 32 |
| 8 | Unpack | Gomez | SCRUM | Python | 32.0 | 30 |
| 6 | Beta | Kanappell | Extreme | Python | 17.8 | 43 |
| 6 | Nester | Kanappell | Extreme | Python | 21.0 | 45 |
| 4 | Alpha | Kanappell | Extreme | Python | 14.0 | 23 |

From the above table we can conclude that the number of people in a team or the team size does not affect the number of escaping defects or we cannot conclude that when the team size increases the no of escaping defects also increases. Because, for example when team size is 14 we have escaping defects to be on the lower side of 45 and also have a higher side of 79 and similarly when team size is 20 we have no of escaping defects to be 73 and 129 respectively.

Also, we have plotted the graph showing team size versus number of escaping defects. From the graph we can conclude that though team size increases, the number of escaping defects does not decrease all the time and it increases at many places and also decreases at many places which makes us conclude that Team Size does not affect the number of escaping defects.

**5. Sorting the data according to the manager who handled the project.**

We took the below data from the given IPC Defect Data for analyzing whether the Managers who handled the projects had an impact on the number of escaping defects or on the product quality.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Manager | Product | Method | Language | Team Size (people) | Size (KLOC) | Escaping Defects |
| Gomez | Zorro | SCRUM | Ruby | 22 | 75.0 | 104 |
| Gomez | Halo | SCRUM | Python | 30 | 103.2 | 98 |
| Gomez | Inspect | SCRUM | Ruby | 24 | 93.2 | 87 |
| Gomez | Mu | Extreme | Python | 18 | 72.2 | 73 |
| Gomez | Storage | Extreme | Python | 20 | 71.0 | 73 |
| Gomez | Guide | Extreme | Python | 12 | 49.3 | 62 |
| Gomez | River | Extreme | Ruby | 18 | 62.0 | 62 |
| Gomez | Quorum | Extreme | Ruby | 16 | 54.6 | 56 |
| Gomez | Yank | SCRUM | Ruby | 16 | 56.0 | 55 |
| Gomez | Lambda | Extreme | Ruby | 10 | 38.0 | 48 |
| Gomez | Precise | Extreme | Python | 14 | 53.3 | 46 |
| Gomez | Value | SCRUM | Ruby | 14 | 50.2 | 45 |
| Gomez | Zebra | SCRUM | Ruby | 14 | 56.0 | 45 |
| Gomez | Kappa | Extreme | Python | 8 | 29.0 | 32 |
| Gomez | Unpack | SCRUM | Python | 8 | 32.0 | 30 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Manager | Product | Method | Language | Team Size (people) | Size (KLOC) | Escaping Defects |
| Kanappell | Zillow | SCRUM | Python | 20 | 67.5 | 129 |
| Kanappell | Window | Extreme | Ruby | 16 | 55.0 | 93 |
| Kanappell | Optimize | Extreme | Ruby | 16 | 51.0 | 92 |
| Kanappell | Xray | SCRUM | Ruby | 14 | 45.0 | 79 |
| Kanappell | Justify | Extreme | Ruby | 12 | 38.1 | 76 |
| Kanappell | Casper | SCRUM | Ruby | 12 | 45.9 | 71 |
| Kanappell | Epsilon | SCRUM | Python | 12 | 44.0 | 61 |
| Kanappell | Zap | SCRUM | Ruby | 18 | 61.0 | 58 |
| Kanappell | Delta | SCRUM | Python | 10 | 36.9 | 56 |
| Kanappell | Zunder | SCRUM | Python | 14 | 49.3 | 54 |
| Kanappell | Framer | SCRUM | Python | 12 | 44.5 | 53 |
| Kanappell | Time | Extreme | Ruby | 10 | 42.0 | 49 |
| Kanappell | Nester | Extreme | Python | 6 | 21.0 | 45 |
| Kanappell | Beta | Extreme | Python | 6 | 17.8 | 43 |
| Kanappell | Alpha | Extreme | Python | 4 | 14.0 | 23 |

|  |  |  |  |
| --- | --- | --- | --- |
| Gomez Manager | Escaping Defects | Kanappell Manager | Escaping Defects |
| **Total** | **916** | **Total** | **982** |

From the above tables, we can conclude that products handled by Manager Gomez had a Total number of escaping defects lesser than the products handled by Manager Kanappell and thus Manager handling the projects had an impact on the number of escaping defects or the product quality.

1. **Sorting the data showing only the products which showed high defects and which the customers complained about.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Product | Manager | Method | Language | Size (KLOC) | Escaping Defects | Critical Defects | Normal Defects | Minor Defects |
| Casper | Kanappell | SCRUM | Ruby | 45.9 | 71 | 14 | 35 | 22 |
| Epsilon | Kanappell | SCRUM | Python | 44.0 | 61 | 12 | 30 | 19 |
| Halo | Gomez | SCRUM | Python | 103.2 | 98 | 20 | 49 | 29 |
| Justify | Kanappell | Extreme | Ruby | 38.1 | 76 | 16 | 38 | 22 |
| Optimize | Kanappell | Extreme | Ruby | 51.0 | 92 | 17 | 45 | 30 |
| Xray | Kanappell | SCRUM | Ruby | 45.0 | 79 | 15 | 40 | 24 |
| Zillow | Kanappell | SCRUM | Python | 67.5 | 129 | 25 | 65 | 39 |
| Zorro | Gomez | SCRUM | Ruby | 75.0 | 104 | 21 | 51 | 32 |

From the above table we analyzed that, the number of escaping defects which is the total of critical, normal and minor defects were on the higher side for these products which the customers complained about. Out of these, most of the products were developed using SCRUM development process and only 2 were done by following Extreme Programming. Thus, the development process had its impact on product quality and for receiving customer complaints.

**4.2 Data Analysis Phase 2: Analyze Release Date and Software Size** to see if either of these correlates to the defect level.

**Pearson Coefficient (Size (KLOC) Vs Escaping Defects):**

|  |  |  |
| --- | --- | --- |
| **x** | **y** | **n = 30** |
| Size (KLOC) | Escaping Defects | Pearson Co-Efficient |
| 14.0 | 23 | 0.104752 |
| 17.8 | 43 | 0.047298 |
| 45.9 | 71 | -0.00273 |
| 36.9 | 56 | 0.007183 |
| 44.0 | 61 | 0.001107 |
| 44.5 | 53 | 0.004653 |
| 49.3 | 62 | 0.00015 |
| 103.2 | 98 | 0.12796 |
| 93.2 | 87 | 0.070702 |
| 38.1 | 76 | -0.01152 |
| 29.0 | 32 | 0.048305 |
| 38.0 | 48 | 0.013909 |
| 72.2 | 73 | 0.014544 |
| 21.0 | 45 | 0.038514 |
| 51.0 | 92 | 0.000132 |
| 53.3 | 46 | -0.00288 |
| 54.6 | 56 | -0.00185 |
| 62.0 | 62 | -0.00099 |
| 71.0 | 73 | 0.013756 |
| 42.0 | 49 | 0.008938 |
| 32.0 | 30 | 0.044366 |
| 50.2 | 45 | 0.000945 |
| 55.0 | 93 | 0.008514 |
| 45.0 | 79 | -0.00658 |
| 56.0 | 55 | -0.00295 |
| 56.0 | 45 | -0.00652 |
| 61.0 | 58 | -0.00373 |
| 67.5 | 129 | 0.076695 |
| 75.0 | 104 | 0.069158 |
| 49.3 | 54 | 0.001054 |
| **Final Pearson Coefficient** | | **0.6629** |
|

**Formula used for Pearson Coefficient:**

= (((size in KLOC-mean of x) \* (escaping defects-mean of y)) / (n\*standard deviation of x \* standard deviation of y)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **mean of x =** | **50.9346667** | **Standard deviation of x =** | **19.9180464** | **n \* Standard deviation of x \* Standard deviation of y =** | **14197.7** |
| **mean of y =** | **63.2666667** | **Standard deviation of y =** | **23.7602005** |  |  |

Pearson Co-Efficient = 0.6629

Since the value of Pearson Co-efficient for Product Size/KLOC and Number of Escaping defects is not equal to +1, it means there is no perfect linear relationship between these two. However, since the value lies between 0 and +1 i.e., it is 0.6629 then it shows that there exists some linear relationship which means when product size increases the number of escaping defects also increases though not following perfect linear relationship.

Spearman Coefficient: (Size (kloc) Vs Escaping Defects).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **x** | **x'** | **x''** | **x'''** | **y** | **y'** | **y''** | **y'''** | **n = 30** |
| Size (KLOC) |  |  |  | Escaping Defects |  |  |  | **Spearman Co-Efficient** |
| 14.0 | 14.0 | 1 | 1 | 23 | 23 | 1 | 1 | **0.09051065** |
| 17.8 | 17.8 | 2 | 2 | 43 | 30 | 2 | 4 | **0.06683367** |
| 45.9 | 21.0 | 3 | 13 | 71 | 32 | 3 | 20 | **-0.004843** |
| 36.9 | 29.0 | 4 | 6 | 56 | 43 | 4 | 14.5 | **0.00408966** |
| 44.0 | 32.0 | 5 | 10 | 61 | 45 | 6 | 17 | **-0.0035515** |
| 44.5 | 36.9 | 6 | 11 | 53 | 45 | 6 | 11 | **0.00871743** |
| 49.3 | 38.0 | 7 | 14 | 62 | 45 | 6 | 18.5 | **-0.0019372** |
| 103.2 | 38.1 | 8 | 30 | 98 | 46 | 8 | 28 | **0.07802642** |
| 93.2 | 42.0 | 9 | 29 | 87 | 48 | 9 | 25 | **0.05521042** |
| 38.1 | 44.0 | 10 | 8 | 76 | 49 | 10 | 23 | **-0.0242151** |
| 29.0 | 44.5 | 11 | 4 | 32 | 53 | 11 | 3 | **0.06188303** |
| 38.0 | 45.0 | 12 | 7 | 48 | 54 | 12 | 9 | **0.02378461** |
| 72.2 | 45.9 | 13 | 27 | 73 | 55 | 13 | 21.5 | **0.02970385** |
| 21.0 | 49.3 | 14 | 3 | 45 | 56 | 14.5 | 6 | **0.05112076** |
| 51.0 | 49.3 | 15 | 17 | 92 | 56 | 14.5 | 26 | **0.00678023** |
| 53.3 | 50.2 | 16 | 18 | 46 | 58 | 16 | 8 | **-0.0080717** |
| 54.6 | 51.0 | 17 | 19 | 56 | 61 | 17 | 14.5 | **-0.0015067** |
| 62.0 | 53.3 | 18 | 24 | 62 | 62 | 18.5 | 18.5 | **0.01097751** |
| 71.0 | 54.6 | 19 | 26 | 73 | 62 | 18.5 | 21.5 | **0.02712091** |
| 42.0 | 55.0 | 20 | 9 | 49 | 71 | 20 | 10 | **0.01539004** |
| 32.0 | 56.0 | 21.5 | 5 | 30 | 73 | 21.5 | 2 | **0.06102204** |
| 50.2 | 56.0 | 21.5 | 16 | 45 | 73 | 21.5 | 6 | **-0.0020448** |
| 55.0 | 61.0 | 23 | 20 | 93 | 76 | 23 | 27 | **0.02227789** |
| 45.0 | 62.0 | 24 | 12 | 79 | 79 | 24 | 24 | **-0.0128071** |
| 56.0 | 67.5 | 25 | 21.5 | 55 | 87 | 25 | 13 | **-0.0064574** |
| 56.0 | 71.0 | 26 | 21.5 | 45 | 92 | 26 | 6 | **-0.024538** |
| 61.0 | 72.2 | 27 | 23 | 58 | 93 | 27 | 16 | **0.00161434** |
| 67.5 | 75.0 | 28 | 25 | 129 | 98 | 28 | 30 | **0.05930008** |
| 75.0 | 93.2 | 29 | 28 | 104 | 104 | 29 | 29 | **0.07264529** |
| 49.3 | 103.2 | 30 | 15 | 54 | 129 | 30 | 12 | **0.00075336** |
|  |  |  |  |  | **Final Spearman Coefficient** | | | **0.65778966** |

**Formula used for Spearman Coefficient: (Size in KLOC Vs Escaping Defects)**

= ((( x’’’- mean of x’’’ )\*( y’’’- mean of y’’’ ))/(n \* standard deviation of x’’’ \* standard deviation of y’’’ )

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **mean of x''' =** | **15.5** | **Standard deviation of x''' =** | **8.802** | **n\*standard deviation fo x''' \* standard deviation of y''' =** | **2322.931** |
| **mean of y''' =** | **15.5** | **Standard deviation of y''' =** | **8.797** |  |  |

Spearman Co-Efficient = 0.6577

Since the value of Spearman Co-efficient for Product Size/KLOC and Number of Escaping defects is not equal to +1, it means there is no perfect linear relationship between these two. However, since the value lies between 0 and +1 i.e., it is 0.6577 here it shows that there exists some linear relationship which means when product size increases the number of escaping defects also increases though not following perfect linear relationship.

**Pearson Coefficient: (Release Date Vs Escaping Defects)**

|  |  |
| --- | --- |
| Rel Date | Escaping Defects |
| 1-Jan-15 | 23 |
| 1-Feb-15 | 43 |
| 1-Mar-15 | 71 |
| 1-May-15 | 56 |
| 1-Jun-15 | 61 |
| 1-Jul-15 | 53 |
| 1-Aug-15 | 62 |
| 1-Oct-15 | 98 |
| 1-Nov-15 | 87 |
| 1-Dec-15 | 76 |
| 1-Jan-16 | 32 |
| 15-Feb-16 | 48 |
| 1-Mar-16 | 73 |
| 1-May-16 | 45 |
| 15-Jun-16 | 92 |
| 1-Jul-16 | 46 |
| 1-Aug-16 | 56 |
| 1-Oct-16 | 62 |
| 1-Nov-16 | 73 |
| 1-Dec-16 | 49 |
| 1-Jan-17 | 30 |
| 28-Feb-17 | 45 |
| 1-Mar-17 | 93 |
| 1-May-17 | 79 |
| 1-Jun-17 | 55 |
| 30-Jul-17 | 45 |
| 1-Aug-17 | 58 |
| 1-Oct-17 | 129 |
| 1-Nov-17 | 104 |
| 1-Dec-17 | 54 |

**Formula used for Pearson Coefficient: (Release Date Vs Escaping Defects)**

= PEARSON((All values of Release Date),(All Values of No. of Escaping Defects))

Or

=CORREL((All values of Release Date),(All Values of No. of Escaping Defects))

Note: Here mean of release date cannot be found. Hence, we have used excel function namely PEARSON() and also CORREL() to determine the Pearson Coefficient.

pearson Co-Efficient = 0.2648

Since the value of Pearson Co-efficient for release date and Number of Escaping defects is not equal to +1, it means there is no perfect linear relationship between release date and Number of Escaping defects. However, since the value lies between 0 and +1 i.e., it is 0.2648 which shows that there exists a very minimal amount of linear relationship between product size and number of escaping defects.

**Spearman Coefficient: (Release Date Vs Escaping Defects)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **x** | **x'** | **x''** | **x'''** | **y** | **y'** | **y''** | **y'''** | **n = 30** |
| **Rel Date** |  |  |  | **Escaping Defects** |  |  |  | **Spearman Co-Efficient** |
| 1-Jan-15 | 1-Jan-15 | 1 | 1 | 23 | 23 | 1 | 1 | 0.0905006 |
| 1-Feb-15 | 1-Feb-15 | 2 | 2 | 43 | 30 | 2 | 4 | 0.06682625 |
| 1-Mar-15 | 1-Mar-15 | 3 | 3 | 71 | 32 | 3 | 20 | -0.02421241 |
| 1-May-15 | 1-May-15 | 4 | 4 | 56 | 43 | 4 | 14.5 | 0.00495009 |
| 1-Jun-15 | 1-Jun-15 | 5 | 5 | 61 | 45 | 6 | 17 | -0.00677947 |
| 1-Jul-15 | 1-Jul-15 | 6 | 6 | 53 | 45 | 6 | 11 | 0.01840143 |
| 1-Aug-15 | 1-Aug-15 | 7 | 7 | 62 | 45 | 6 | 18.5 | -0.01097629 |
| 1-Oct-15 | 1-Oct-15 | 8 | 8 | 98 | 46 | 8 | 28 | -0.04035401 |
| 1-Nov-15 | 1-Nov-15 | 9 | 9 | 87 | 48 | 9 | 25 | -0.02657984 |
| 1-Dec-15 | 1-Dec-15 | 10 | 10 | 76 | 49 | 10 | 23 | -0.01775577 |
| 1-Jan-16 | 1-Jan-16 | 11 | 11 | 32 | 53 | 11 | 3 | 0.02421241 |
| 15-Feb-16 | 15-Feb-16 | 12 | 12 | 48 | 54 | 12 | 9 | 0.00979257 |
| 1-Mar-16 | 1-Mar-16 | 13 | 13 | 73 | 55 | 13 | 21.5 | -0.00645664 |
| 1-May-16 | 1-May-16 | 14 | 14 | 45 | 56 | 14.5 | 6 | 0.00613381 |
| 15-Jun-16 | 15-Jun-16 | 15 | 15 | 92 | 56 | 14.5 | 26 | -0.00225982 |
| 1-Jul-16 | 1-Jul-16 | 16 | 16 | 46 | 58 | 16 | 8 | -0.00161416 |
| 1-Aug-16 | 1-Aug-16 | 17 | 17 | 56 | 61 | 17 | 14.5 | -0.00064566 |
| 1-Oct-16 | 1-Oct-16 | 18 | 18 | 62 | 62 | 18.5 | 18.5 | 0.00322832 |
| 1-Nov-16 | 1-Nov-16 | 19 | 19 | 73 | 62 | 18.5 | 21.5 | 0.0090393 |
| 1-Dec-16 | 1-Dec-16 | 20 | 20 | 49 | 71 | 20 | 10 | -0.01065346 |
| 1-Jan-17 | 1-Jan-17 | 21 | 21 | 30 | 73 | 21.5 | 2 | -0.03196038 |
| 28-Feb-17 | 28-Feb-17 | 22 | 22 | 45 | 73 | 21.5 | 6 | -0.02657984 |
| 1-Mar-17 | 1-Mar-17 | 23 | 23 | 93 | 76 | 23 | 27 | 0.03712569 |
| 1-May-17 | 1-May-17 | 24 | 24 | 79 | 79 | 24 | 24 | 0.03109949 |
| 1-Jun-17 | 1-Jun-17 | 25 | 25 | 55 | 87 | 25 | 13 | -0.01022302 |
| 30-Jul-17 | 30-Jul-17 | 26 | 26 | 45 | 92 | 26 | 6 | -0.04293667 |
| 1-Aug-17 | 1-Aug-17 | 27 | 27 | 58 | 93 | 27 | 16 | 0.00247505 |
| 1-Oct-17 | 1-Oct-17 | 28 | 28 | 129 | 98 | 28 | 30 | 0.07801776 |
| 1-Nov-17 | 1-Nov-17 | 29 | 29 | 104 | 104 | 29 | 29 | 0.0784482 |
| 1-Dec-17 | 1-Dec-17 | 30 | 30 | 54 | 129 | 30 | 12 | -0.02184497 |
|  |  |  |  |  | **Final Spearman Coefficient** | | | **0.17841854** |

**Formula used for Spearman Coefficient: (Release date Vs Escaping Defects)**

=(( ( x’’’- mean of x’’’ ) \* ( y’’’- mean of y’’’ )) / (n \* standard deviation of x’’’ \* standard deviation of y’’’)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **mean of x''' =** | **15.5** | **Standard deviation of x''' =** | **8.803408** | **n\*standard deviation of x''' \* standard deviation of y''' =** | **2323.189** |
| **mean of y''' =** | **15.5** | **Standard deviation of y''' =** | **8.796551** |  |  |

Spearman Co-Efficient = 0.1784

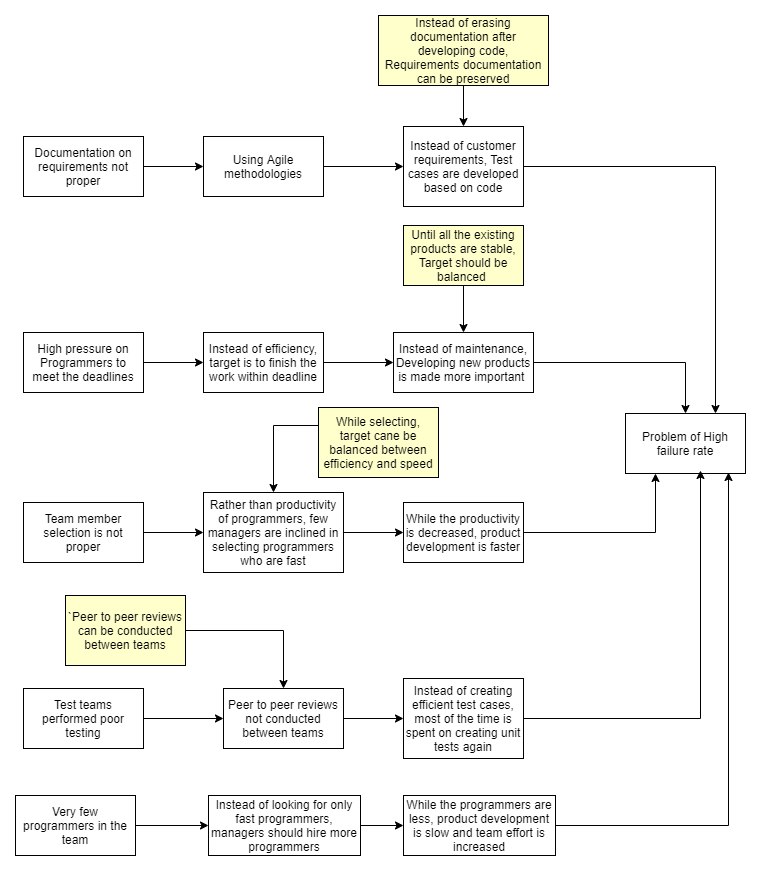
Since the value of Spearman Co-efficient for release date and Number of Escaping defects is not equal to +1, it means there is no perfect linear relationship between release date and Number of Escaping defects. However, since the value lies between 0 and +1 i.e., it is 0.1784 which shows that there exists a very minimal amount of linear relationship between product size and number of escaping defects.

| Table 4.2 – Summary of Phase 2 Data Analysis  Identifying Correlations Related to High Defect Rate Problem | | |
| --- | --- | --- |
| Data Analysis Method | Description (what data were used for each analysis, what refinement, calculations, and analysis methods were used) | Conclusions reached from this analysis method (including methods that resulted in no conclusion) |
| Pearson Co-Efficient | Product Size (KLOC) and No of Escaping Defects where used for calculating the Pearson coefficient.  Since there are 30 products here n = 30. Also, x refers to product size (KLOC) and y refers to no of escaping defects.  Formula:  Pearson Coefficient  = (((Product size in KLOC-mean of x) \* (escaping defects-mean of y))/ (n\*standard deviation of x \* standard deviation of y) | Since the value of Pearson Co-efficient for Product Size/KLOC and Number of Escaping defects is not equal to +1, it means there is no perfect linear relationship between these two. However, since the value lies between 0 and +1 i.e., it is 0.6629 then it shows that there exists some linear relationship which means when product size increases the number of escaping defects also increases though not following perfect linear relationship. |
| Spearman Co-Efficient | Here the release date and No of Escaping Defects (actual) where taken and rank was assigned to both and their corresponding Rank of release date and Rank of No of Escaping Defects where used for calculating the Spearman coefficient.  Since there are 30 products here n = 30. Also, x’’’ refers to rank of product release date and y’’’ refers to rank assigned to the no of escaping defects for each product.  Formula:  Spearman Coefficient: (Rank of Release Date Vs Rank of No of Escaping Defects)  = ((( x’’’- mean of x’’’ )\*( y’’’- mean of y’’’ ))/(n \* standard deviation of x’’’ \* standard deviation of y’’’ ) | Since the value of Spearman Co-efficient for release date and Number of Escaping defects (rank values for both were taken into account) is not equal to +1, it means there is no perfect linear relationship between release date and Number of Escaping defects. However, since the value lies between 0 and +1 i.e., it is 0.1784 which shows that there exists a very minimal amount of linear relationship between product size and number of escaping defects. |
| Pearson Co-Efficient | Here the Release Date and No of Escaping Defects (actual) where taken for calculating the Spearman coefficient.  Formula:  Pearson Coefficient (Release Date Vs No. Of Escaping Defects)  = PEARSON((All values of Release Date),(All Values of No. of Escaping Defects))  Or  =CORREL((All values of Release Date),(All Values of No. of Escaping Defects)) | Since the value of Pearson Co-efficient for Release Date and Number of Escaping defects is not equal to +1, it means there is no perfect linear relationship between these two. However, since the value lies between 0 and +1 i.e., it is 0.2648 here it shows that there exists very minimal linear relationship exists. |
| Spearman Co-Efficient | Here the Product Size (KLOC) and No of Escaping Defects (actual) where taken and rank was assigned to both and their corresponding Rank of Product Size (KLOC) and Rank of No of Escaping Defects where used for calculating the Spearman coefficient.  Since there are 30 products here n = 30. Also, x’’’ refers to rank of product size (KLOC) and y’’’ refers to rank assigned to the no of escaping defects for each product.  Formula:  Spearman Coefficient: (Rank of Product Size in KLOC Vs Rank of No of Escaping Defects)  = ((( x’’’- mean of x’’’ )\*( y’’’- mean of y’’’ ))/(n \* standard deviation of x’’’ \* standard deviation of y’’’ ) | Since the value of Spearman Co-efficient for Product Size/KLOC and Number of Escaping defects (Here rank is taken into account) is not equal to +1, it means there is no perfect linear relationship between these two. However, since the value lies between 0 and +1 i.e., it is 0.6577 here it shows that there exists some linear relationship which means when product size increases the number of escaping defects also increases though not following perfect linear relationship. |

4.3.a Root Causes of High Defect Rate Problem:

| Table 4.3.a – Root Causes of High Defect Rate Problem | | | |
| --- | --- | --- | --- |
| Root Cause | Description | Importance (High, Medium, Low) | Reasons for indicated level of Importance |
| High Pressure on programmers to meet deadlines | Instead of efficiency, more focus is put on programmers to finish the work within deadline. Instead of maintenance of existing products, Developing new products is made more important. | High | Quality of product is not taken into consideration. Poor quality of the product may cause dissatisfaction to customer. Quality is a high priority. |
| Test teams have performed poor testing | instead of concentrating on productivity of programmers, few managers are inclined in selecting programmers who are fast. The productivity is decreased but product development is faster. Peer to peer reviews are not conducted between teams. | High | Without proper testing, the software product delivered may cause failure or a lot of defects which will consequently lead to customer distrust and dissatisfaction. Good testing of product is of high priority. |
| No proper documentation | Customer requirements are to be considered for preparing test cases. but test cases are developed based on code instead. Use of agile methodologies need more documentation for continuity of workflow. Some parameters of product development cannot be remembered. | Medium | Considering customer requirements is a priority. test cases developed only on code may miss out certain boundary test conditions. |
| No proper team member selection | Productivity of programmers is more important for creating a product with good quality. Managers should concentrate on choosing productive programmers rather than hiring programmers who are fast in coding. Fast programmers may cause a lot of errors. But the product development is faster. | Medium | Having a good team will help build team co-operation and productivity is increased. The right balance of efficiency, speed and skilled people in the team is needed. This makes the team member selection of medium priority |
| Few programmers in the team | Instead of looking for only fast programmers, managers should hire more programmers. While the programmers are less, product development is slow and team effort is increased | Low | Having more members in the team will reduce the total effort of each programmer. this may increase productivity and reduce pressure from the programmers. Increasing the number of programmers is optional and of low priority. |

4.3.b Causal Map:



**4.4 Collect Data:** To be performed, if necessary, to identify most critical root causes.

**IMPROVEMENTS and RECOMMENDATIONS**

**5.1 Most Important Root Causes:**

| Table 5.1.a – Old school company policies are still followed |
| --- |
| The company policies made for the programmers are not favorable. Programmers not allowed to do multitasking. If there are two or more projects, programmers cannot work on all of them together. There is no availability of programmers that are suitable and best.  No availability of all CCB members at the same time. Approval of repairs were getting delayed. Signatures of CCB members to start the work were getting delayed.  There is delay in assigning the tasks and a delay in fixing the problem.  All the above problems arise because of following old school policies in the company. |

| Table 5.1.b – Poor Documentation |
| --- |
| Customer requirements are to be considered for preparing test cases. but test cases are developed based on code instead.  Use of agile methodologies need more documentation for continuity of workflow. Some parameters of product development cannot be remembered.  Considering customer requirements is a priority. test cases developed only on code may miss out certain boundary test conditions.  Whenever a person left the company or another person is replaced, the person is left clueless about the work due to the lack of documentation.  When the company used agile methodology, tracking the stories and sprints were difficult due to bad documentation. |

| Table 5.1.c – Higher priority for development Methods |
| --- |
| The managers gave higher Priority to development rather than maintenance. The project followed Agile Methodology and did not document the stories and sprints properly. This made the programmers confused and they did not have a continuity through their development. Documentation is very less or lack of Documentation.  Programmers who wrote the code not able to understand their own code. Assumptions made by maintenance programmer may go wrong and required rework.  The development methods used were more important than Quality of the product. The use of Scrum or Xtreme programming would increase the productivity but lack of documentation would ruin the maintenance activities. |

| Table 5.1.d – Old Technology |
| --- |
| The use of old technology decreased the productivity of the programmers and they were not able to build a product with good quality.  The old technologies used hindered the software product development. It reduced the speed and efficiency and programmers could not deliver the desired results. It slowed the entire development process.  The projects delivered used outdated old technologies that were difficult to be maintained and required programmers who still knew how the technology works. |

| Table 5.1.e – High pressure on Programmers to meet deadlines |
| --- |
| Instead of efficiency, target was to finish the work within deadline. Instead of maintenance, Developing new products was made more important. Until all the existing products are stable, Target should be balanced.  There was high pressure for the programmers to complete the tasks on time. The documentation part was ignored. The maintenance requests were taking the back seat. Development of new products and delivering them was of high priority. |

**5.2 IMPROVE:**

**Recommended Solutions:**

Poor Documentation:

Instead of putting effort on meeting the deadlines, the managers should insist the programmers to do proper documentation. This in turn will help the project for the maintenance activities. Documentation also helps the project while following agile methodologies for sprints and stories. Programmers will be able to identify what they have coded themselves with the help of documentation and also the same will help if the programmer is replaced or if other programmers work on this project/Product.

Old company policies:

Programmers should be allowed to do multitasking. If there are two or more projects, programmers should be allowed to work on multiple projects at the same time. This Followed along with proper documentation by programmers will allow them to manage multiple projects at the same time. If there is multitasking, there will not be any delay in assigning tasks to the programmers and the defects will not be missed hence resulting in lower defect count.

Focus on new product development:

The company should change its focus on both development of new products and give more importance to maintenance of existing products as they are used by the customers in production and may have high financial impact.

Longer wait from CCB:

If there is a change in company policy, eventually there will be availability of all CCB members at the same time. The approval of repairs by the CCB members will not get delayed. The signatures of CCB members to start the work will not get delayed. Productivity will be increased and the team will be able to meet all the deadlines. The ccb team members should discuss and come to a consensus to fix a schedule for approving the pending change requests on a daily basis.

Other general improvements:

Since the maintenance people complain about the lack of unit testing and system testing done by the developers, proper budget and time should be allocated to carry the testing. This results in reduced defects and improved product quality.

Also, the company policy should be reviewed and clauses which are appropriate to them should be amended. In order to make the process more efficient in-person meetings should be replaced with electronically approved emails to speed up the process. Most importantly, the quality assurance should be performed to review the overall testing process being followed and the programmers must be kept informed about rising number of defects in few products.

**Test the Solutions:** To be performed

**Assess Risks of Implementation:** To be performed

**Create full implementation plan:** To be performed

**Deploy the plan:** To be performed

**CONTROL:**

**Control Plan:** To be performed

**Monitor and Control:** To be performed

**Update training, process documents, procedures, etc.:** To be performed

**Develop Response Plan:** To be performed