ASSIGNMENT 4

4

**DMAIC PLAN / STATUS**

CSE 6329 -- SOFTWARE MEASUREMENT AND QUALITY ENGINEERING

Professor Dennis J. Frailey

**Spring, 2018**

|  |  |
| --- | --- |
| NAME(s):**Karthik Somanahalli Muralidhara** | ID Number(s): **1001520125** |
| **Subrahmanya Basavapatna Nagaraja Rao** | **1001540208** |

|  |  |  |  |
| --- | --- | --- | --- |
| **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** | **Identify All Organizations** |  |
| **/ 8** | **Identify All Process Steps** |
| **/ 4** | **Other Info** |
| **<total goes here>** | 1. **Analyze -- Root Cause Analysis (25 points)** | | |
| **/ 10** | **Technique 1** |  |
| **/ 10** | **Technique 2** |
| **/ 5** | **Root Cause(s)** |
| **<total goes here>** | 1. **Analyze -- Causal Model (20 points)** | | |
| **/ 15** | **Identify All Possible Causes** |  |
| **/ 5** | **Relationships Make Sense** |
| **<total goes here>** | 1. **Analyze -- Most Important Root Causes / Flow Diagrams (15 pts)** | | |
| **/ 5** | **Cause 1** |  |
| **/ 5** | **Cause 2** |
| **/ 5** | **Cause 3** |
| **<total goes here>** | 1. **Improve -- Recommendations (10 points)** | | |
| **/ 6** | **Good 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. GAMMA, one of our most important customers, is also one of the ones complaining the most about this problem, although we are losing other customers because of this. We must correct this in order to satisfy our customers, especially GAMMA.

**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.

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

**Timeline:** We must resolve this by the end of the year.

**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 problem:** Slow response to software failures and higher failure rates.

**The customers:** Several, notably GAMMA Corporation.

**Voice of the customer:** See memo from GAMMA. Customer quality requirement is software that runs properly, with minimal failures, and rapid response to correcting the software when it does fail.

**CTQs:** Measurable attributes that are critical to quality for this customer.

| CTQ #1 | How Measured | Why it is Critical |
| --- | --- | --- |
| **Policy that a programmer once assigned to a particular problem or failure, cannot be pulled off or multitask on another problem , unless he corrects that failure.** | **1)** Number of Programmers working on a specific problem/ new problem.  **2)** Number of problems yet to be assigned to a programmer. | This CTQ is critical because a programmer cannot be **withdrawn from a task**, even though it is unimportant or easy, in order to perform a **high prioritized task**. This causes unwanted delay and hence affects the overall quality of the software. This policy was designed by a **former Vice President**, which is still in force in the company. |

| CTQ # 2 | How Measured | Why it is Critical |
| --- | --- | --- |
| **Delay in waiting time occurs for Regression Testing if there are one or more modules being fixed additional to the module being fixed, on the same software.** | **1**) Time Taken for Regression Testing.  **2)** Number of modules being fixed before Regression Testing is done. | Regression testing is running a **large suite of tests** on the complete**, final software package**, with the repaired modules included in place of the ones that had errors. The regression test software is the only software which is given from the original program development team, and sometimes it has to be adjusted, because new features would have been added since then. Hence fixing other modules in a software, while regression testing is performed on it, can cause a **several-week delay** in **final approval** of a repaired module. |

**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 organization’s process may be subject to improvement based on findings from our analysis. Specific processes needing improvement are defined as a result of the analysis.

**Project Targets:** Reduce response time by at least 50%; reduce failure rate.

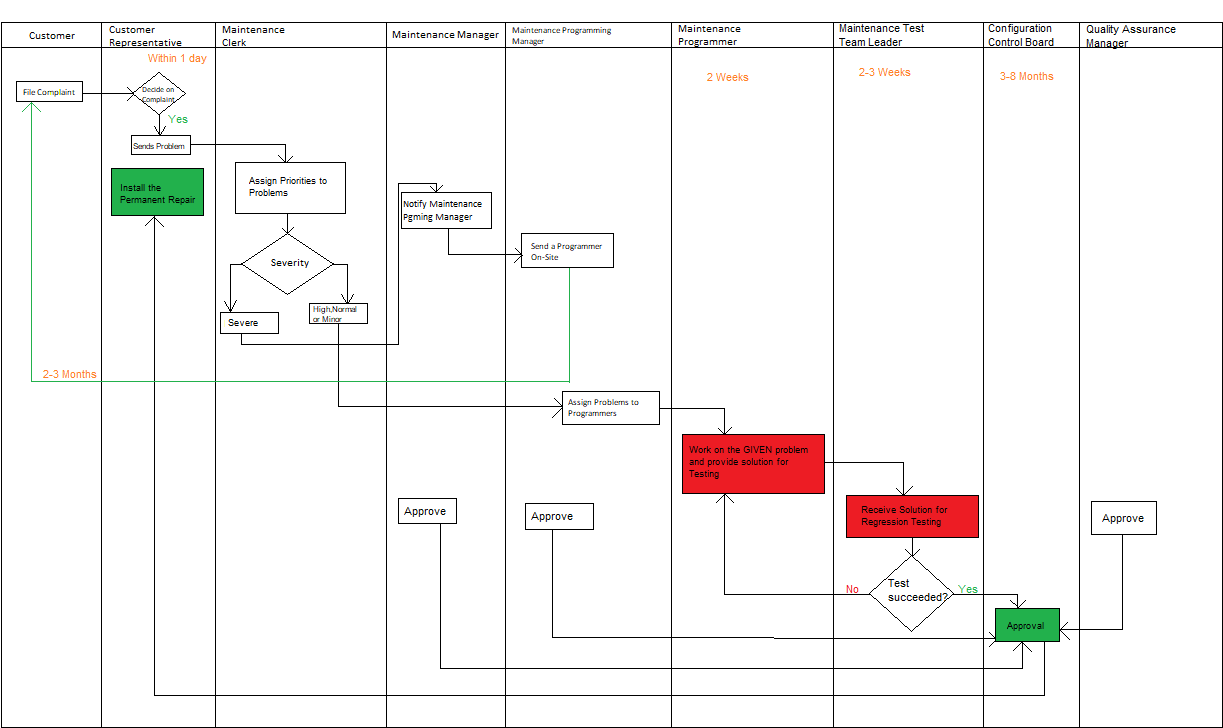
**MEASURE**

**Organizations/People (Roles) Involved:**

|  |  |
| --- | --- |
| **Organization/Role** | **Summary of what it does** |
| Girisha Gotti ( **Information Processing Manager** of **ACME**) | Submits a statement of complaint to Customer Rep, James Johnson. |
| James Donohu ( **President** and **CEO** of **IPC**) | Chief executive officer of IPC. Handles all operations to run efficiently at IPC. |
| Wally Ott (**Executive Assistant**) | Provide high-quality administrative and clerical assistance to top-level executives. |
| Rachel Wallace ( **Software Development Manage**r of **IPC**) | Supervise all software development for new products |
| Paul Watson ( **Marketing and Sales Manager of IPC**) | Accomplish **marketing and sales** objectives by planning, developing, implementing, and evaluating advertising, merchandising, and trading promotion programs. |
| Jeff Arterbum (**Quality Assurance Manager** of **IPC**) | Supervise QA staff and serves on both software development CCB and software maintenance CCB. |
| Melinda Shah (**Maintenance Manager** of **IPC)** | Resolve priority disputes with problem reports, to assure that there are adequate people and resources for testing of all repairs, and to assure that everything is ready before shipping updates back to the customer reps. Also a member of maintenance CCB and Supervisor to George Wilson. |
| James Johnson ( **Customer Representative**) | IPC’s representative to some of its major customers, including **ACME, Delta**, and **Zeta** **Corporation**. |
| Walter Weston ( **Customer Representative**) | IPC’s representative to some of its major customers |
| George Wilson ( **Maintenance Clerk** of **IPC**) | Review incoming problem reports, prioritize them, combine similar reports that probably require the same fix, and ship updates back to the customer rep. |
| Sharleen Jefferson (**Maintenance Programming Manager** of **IPC** ) | Supervise the programmers and assigns them to work on specific problem report repairs. Also sits on the CCB. |
| Wendy Stottlemeyer (**Maintenance Test Team Leader** of **IPC** ) | Report to Melinda (Maintenance Manager) and perform 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. |
| Narayan Bhat ( **Maintenance Programmer** of **IPC** ) | Works for Sharleen, the Maintenance Programming Manager. Analyze problem reports, identify the source of the problem and make the required repairs. |

**Process Flow:**

The Swim Lane Diagram is shown below. At first, The **Customer**(Girisha) files a complaint to the **Customer Represntative**(James). The CR then extracts relevant data from the file complaint and sends the problem report to the **Maintenance Clerk** (George). The Maintenance Clerk then assigns **priorities** to the problems received as Severe, High, normal or minor. If it is a Severe Problem, then the **Maintenance Manager** ( Melinda) is consulted, who notifies the Maintenance Programming Manager (Sharleen) to **send a programmer on-site** to repair the problem. If the priority is other than severe, then it is directly sent to the **Maintenance Programming Manager**(Sharleen), who assigns the given problems to the Maintenance Programmers ( Narayan) respectively. The programmer works on the given problem(Can only work on ONE given problem which is the 1st CTQ here), and then sends the solution to the Testing Team for Testing. The **Maintenance Test team Leader** (Wendy) receives the solution from programmers and subjects it to **Regression Testing**. Regression testing is running a **large suite of tests** on the complete**, final software package**, with the repaired modules included in place of the ones that had errors. If there are **one or more modules** being fixed during Regression Testing, additional to the module being fixed, the regression test **may fail** and will be sent back to Maintenance Programmer. This causes **unwanted delay** (2nd CTQ). If the test succeeds, then the final repair will be kept pending for **Approva**l from **Configuration Control Board (CCB)**, consisting of Maintenance Manager(Melinda), Maintenance Programming Manager(Sharleen) and Quality Assurance Manager(Jeff). Once approved, the final repair package will be sent to the Customer Representative(James) who installs the repair package on-site.

****

**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 (from IPC Data Report)

**Collect Data:** Interview results have been collected and are summarized in various supplementary files.

**ANALYZE:**

**Root Cause Analysis:**.

We have used 2 root cause analysis techniques and they are

1. 5 whys
2. Cause and Effect chart

The 5 whys approach:

1. why is that software failures are occurring more often than in the past and why these problems are taking too long to fix?

a.1 why is that software failures are occurring more often?

-->This is due to more suitable programmers with suitable expertise not being assigned to the more appropriate problems.

a.1.1 why is it that suitable programmers may not be assigned with problems dealing with areas of their expertise?

--> This is due to a policy of the company that 1 programmer must not be assigned with more than 1 problem.

a.1.2 Why is the policy stated as above?

--> The policy believes that a programmer cannot/should not multitask and only when he/she finishes the assigned issue, can he/she be assigned with new problems.

A.1.3 Why cannot a programmer multitask or more appropriately, why does the policy of the company believes that a programmer should not multitask?

🡪 this is because it was believed to be true by the president of the company with his experience which was 8 years ago.

a.5 why did the president believe so?

🡪 since its already old enough to be changed and the president is not working with the company anymore, this can be considered as a root cause.

root cause 1 -> according to the policy, no programmer can be assigned with more than 1 problem even if that programmer is having an expertise to deal with the new problem. The policy of the company to not assign employees with concurrent tasks and not asking them to multitask is one of the key root causes that we found from our analysis in a.5. we firmly believe that the policy was suited to address problems of older age and since the technology and the industry has grown rapidly since, the process is outdated and needs an immediate attention.

==================================================================

a.2 Why is it that these problems are taking too long to fix?

🡪 One of the main reasons might be that the problems are not assigned to programmers as soon as they should have been.

a.2.1 why is the problems not assigned quickly ?

🡪 this is because they are not able to complete the previously assigned problems fast enough.

a.2.2 why are the programmers not able to finish those previously assigned tasks soon enough?

🡪 this is because they are not able to fix the problems sooner.

a.2.3 why are the programmers not able to fix the problems sooner?

🡪 this is because they often have to wait for original programmers who in turn are not readily available to meet since they are busy with their own works.

a.2.4 why should they meet the original programmers and not solve problems on their own?

🡪 this is because they are not able to understand the problem and hence the cannot understand the code better.

a.2.5 why cannot the programmers understand the problem better?

🡪 This is because they have not got the necessary documentation related to the scenario.

a.2.6 why not enough documentation is provided?

🡪 Because the policy was changed 8 years ago and the process was changed to “agile” in which employees don’t “feel” the importance of documentation.

a.2.7 why is documentation required?

🡪 1. so that there is a better understanding of the problem.

2.so that programmers can work independently.

3.so that programmers can start the work on their own and not depend upon the original programmers for meeting regarding the solution.

as we can see, we found another root cause rc2 in a.2.7.

root cause 2 🡪 Lack of enough documentation results in complexities with respect to the understanding of the problem independently and this hence, results in the delay of the problems being fixed.

Causal Map:

<Task 1.4 causal map goes here. It should take approximately one or two pages.>

A policy established years back by a president no longer working for company.

The causal map is as follows

Suitable programmers not being assigned to their areas of expertise.

1. why too many failures?

No real evidence, so this assumption refutes.

Lack of infrastructure?

“Waiting to be assigned” seems to take a longer time

Queuing delays of problems results in the delay of fixing of older ones

Yes, theres been a change to agile process resulting in lower documentation and higher dev efficiency.

Change in development process?

As we can see, we get 3 root causes for the problem described. Priorities of the three are

1.An outdated policy established years ago.

2.Lack of documentation hinders the progress of programmers.

3.The queuing delay.

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

**Prioritize Root Causes:** Below is information on each of the three most important root causes, including a flow diagram of each of the three target sub-processes that must be improved to fix the three root causes.

| Root Cause /Target Sub-process 1: |
| --- |
| <Task 1.5a description of root cause, descriptive name of sub-process, and flow diagram. Take as much space as necessary>  root cause:  An outdated policy established years ago by an employee who is not working with the organization anymore.  Description:  -This root cause is found by the application of 5 whys principle.  -The policy states that no programmer can be assigned with more than 1 problem at any given time.  -This policy causes the programmers NOT to multitask although they might be well capable of it.  - This results in a wastage of time if the currently assigned programmer is having a hard time fixing that problem  -It can also take a significant amount of time if the whole product is under regression tests or any such processes.  -We firmly believe that the policy is outdated and must be improved.  -1 quick suggestion can be the usage of an effective bug tracking system for example JIRA, which allows a programmer to be assigned with multiple problems.  Flow diagram for the 1st root cause is as follows.    -We firmly believe that a programmer can be assigned with at most 3 bugs at a given time and track the status of the solutions regularly.  -if the progress is halted, call up for a 1 on 1 meeting with the manager.  -repeat this process.  Programmer working on a bug?  Busy with the problem?  Yes  no    Track Status  Continue |

|  |
| --- |
| Root Cause /Target Sub-process 2: |
| Root cause 2 is as follows  Lack of documentation hinders the progress of programmers Lack of documentation hinders the progress of programmers.  Description of the root cause is as follows:  -The development process was changed to “agile” 8 years back.  -This was done in order to improve the efficiency of development.  -This resulted in an increase of the efficiency of the development team by 25%.  -It has been observed that since the inception of that process, the documentation of the scenarios and processes has been gradually reduced to a great extent.  -Since there is no documentation, it has hence caused a cascading effect , where the employees are finding it real hard to understand the problem and hence taking a lot of time to fix it.  -This in turn has increased the time take by each programmer to fix a problem.  The flow chart to this process is as follows.  Assign a bug  Programmer knows solution?  no  yes    Continue with the execution  INDEFINITE WAIT for original programmers!!!  We believe that, a solution for an unknown/foreign problem must always be documented.  By documentation, we mean, 1.the client who has raised the problem  2.what is the problem about.  3.solution for the problem with enough details. |

|  |
| --- |
| Root Cause /Target Sub-process 3: |
| <Task 1.5c description of root cause, descriptive name of sub-process, and flow diagram. Take as much space as necessary>  root cause 3 is as follows:  The queuing delay.  The description of the problem is as follows:  -The review of the problem is done by George Wilson .  -The problems are then linked to another older problem if the newer problem is similar.  -This new linked problem gets a new problem id.  -If this problem, at a later stage is found out that its NOT the part of a parent problem , then its delinked and is stood alone with a new id and queued afresh .  -This queuing of the delinked problem is set to that day when it was found that the problem is not part of the parent problem rather than to the day it was actually raised.  -This results in a significant delay to fix that problem.  Flow diagram is as follows:    New Problem arrives  can be linked?  Link it!  Create a new id and queue it.  Is the link correct?  NO    yes  continue  Create a new id and queue it. |

**IMPROVE:**

**Create Solutions:**

The 1st major impact is done by the root cause 1. The proposed solution is as follows and in a flowchart.

programmer working on a bug

Track the status of the fix.

busy with the assigned problem?

YES

no

Assign a new bug.

Is the count of the bug=3

no

The main idea is to assign a new bug for the programmer until the bug count reaches 3.

The second main impact is done by the root cause 2 and the proposed solution is as follows and in a flowchart. The main idea is to document everything that is new/unknown to the organization.

Assign a bug

Programmer knows the solution?

NO Yes

Continue to track.

Search for documents

Found?

yes

NO

Document it afresh

The 3rd most important impact is done by RC number 3.

The solution is as follows

Whenever a new problem arrives, Discuss the solution with the rest of the programmers and have a meeting to know if it can be linked .

Brainstorming can be a really good technique.

**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