ASSIGNMENT 4

4

**ROOT CAUSE ANALYSIS REPORT**

CSE 6329 -- SOFTWARE MEASUREMENT AND QUALITY ENGINEERING

Professor Dennis J. Frailey

**Fall, 2015**

|  |  |
| --- | --- |
| NAME | COURSE SECTION (TUES OR FRI) |
| **Tasmeen kaur kathuria** | **Tuesday** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Grading Comments (student – do not write inside this box)** | | | |
| **<total goes here>** | **Swim Lane Diagram (25 points)** | | |
| **/ 10** | **Identify All Organizations** |  |
| **/ 10** | **Identify All Process Steps** |
| **/ 5** | **Other Info** |
| **<total goes here>** | **Causal Model (20 points)** | | |
| **/ 15** | **Identify All Possible Causes** |  |
| **/ 5** | **Relationships Make Sense** |
| **<total goes here>** | **Root Cause Analysis (40 points)** | | |
| **/ 15** | **Technique 1** |  |
| **/ 15** | **Technique 2** |
| **/ 10** | **Root Cause(s)** |
| **<total goes here>** | **Report (15 points)** | | |
| **/ 10** | **Good Recommendations** |  |
| **/ 5** | **Legibility, Correct English, etc.** |
| **<total>** | **Grand Total** | | |

**Root Cause Analysis Report**

1. **Introduction and Summary**
   1. **Purpose of This Report**

This report shows the results of analyzing IPC’s software maintenance process, as a result of complaints from major customers that it is taking too long to respond to failures encountered in the customer’s environment.

**Approach Taken**

In order to analyze the situation, we have created two structures to help us understand the situation:

* A swim lane diagram outlining the process and the organizations involved in each step (in order to show the overall timeline and the processes contributing to that timeline). This diagram is also used to determine places in process where there are delays, work task that do not add value or that constitute unnecessary rework, excessively complex processes, and other factors that affect quality , cost and cycle time.
* A causal model showing potential causes and contributing factors to the problem.

We then applied root cause analysis techniques to narrow down the contributing factors and, ultimately, to reach a conclusion regarding the root cause(s). This is followed by our recommendations.

* 1. **Root Cause**

Based on the analysis there are several findings regarding this event which are

1. There is excess work in progress because of long waiting queue which is in, turn the result of company policies that restricts the programmer to work on one problem at a time, constraint CCB members to meet once a week, Unlinked problems goes to queue with date it is unlinked rather than the date it was linked and allows regression test to be done only when all current repairs on a particular problem are complete and once in every few weeks.
2. Due to the Agile Shift there is lack of documentation. This results in maintenance organization consulting original software developers. Also, all the unit tests produced by software developers are destroyed furthermore there is no requirements specification documentation which leads to delays. Furthermore, since there is a lack of documentation, maintenance programmer usually overlooks the problem that results to the failure of regression tests. It seems that CEO thinks that they should make more profit on new software products and shifting to Agile reduced their cost but on another hand it led to delays in maintenance processes.

Hence, there are two root causes

1. CEO is more inclined towards improving the productivity of software development group and agile shift reduced their cost significantly.
2. Old Company policies.
   1. **Recommendations**

Based on the analysis, the first root cause is that CEO is more inclined towards improving the productivity of software development group and they reduced their cost significantly by switching to agile technologies. Shifting to Agile proved to be a positive change for new software development team, but it did not turn fruitful for maintenance team. Due to shift to Agile, there is a lack of documentation, unit tests are destroyed and there is no requirements specification document. If development team could, at least, archive the unit tests, make requirements specifications document it would save a lot of time for maintenance programmers. This root cause has the biggest impact on delay.

Another root cause is Company policies that were instituted long ago. With advancement in technologies available, these policies should be changed. CCB members seem to meet once a week and if someone is not able to go for a meeting he/she sends the deputy, and also for final approval all three should sign the approval paperwork on the same day. This restriction should be removed and technologies like skype, emails, and smartphones should be used so that they can talk whenever they like to. Another restriction instituted is that no programmer should multitask. Maintenance programming manager claims that maintenance programmers can multitask well enough to handle more than one assignment if they are allowed to do so. So I suggest this restriction should be removed as it leads to a lot of delays. Also, there is a prioritization problem. There should be monitoring of the distribution of priorities throughout the process also the number of priority tasks should be restricted.

1. **Customer Complaints**

On October 15, 2015, Lars Jhonson, Information technology manager of Acme Corporation wrote to Mr. James Donohu, President of IPC regarding the time it takes to repair the problems. Mr. Lars Jhonson complaints that it takes 4 months to install the software that corrects the corresponding errors and this level of response is unacceptable. He also requests to contact him immediately to discuss how IPC is addressing this problem. Also, if there is no significant improvement by the end of 2015 Acme Corporation would not renew their contract with IPC.

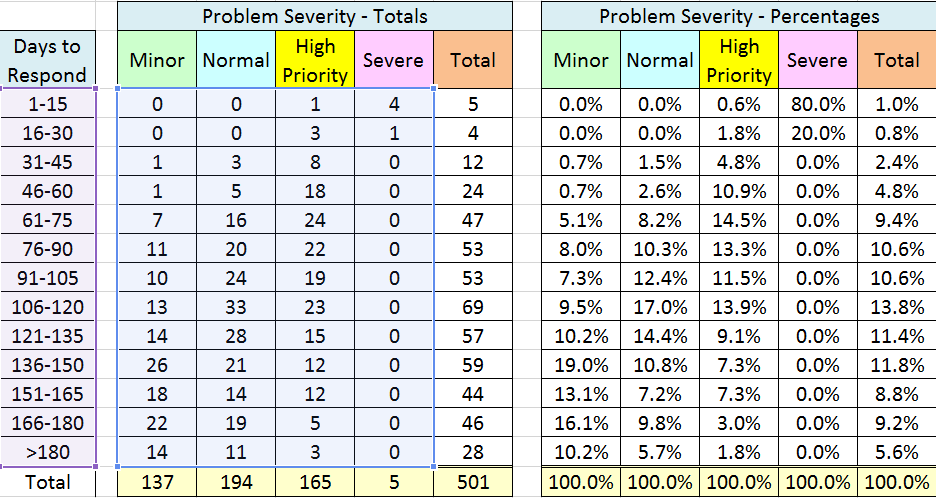
It is the 7th such customer complaint and IPC might lose several valuable customers.

It can be noted that at an average it takes around 15 days for a severe problem to get solved, 2-3 months for a high priority task to get solved, normal priority tasks may take 4-5 months and low priority can take up to 8 months.

According to the analysis done

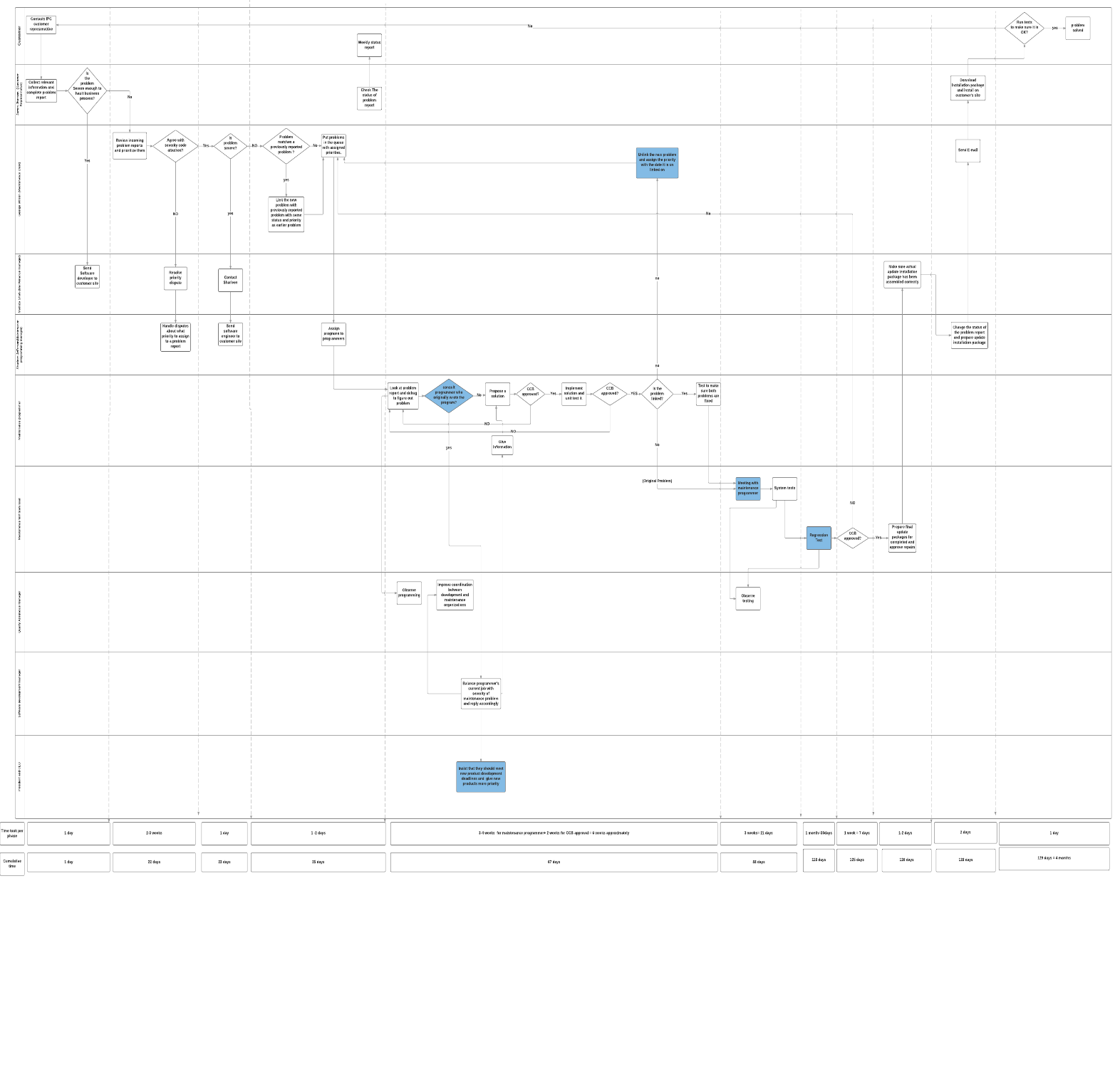
* from 501 total complaints 14% of the total problems take 4 months,
* from 165 of high priority complaints approximately, 15% problems take 61-75 days and 14% problems take 120 days,
* from 194 of normal priority problems 15%-20% problems takes 106 to 120 days, and
* From 137 of minor priority problems 15%-20% problems takes 136-150 days.
* From 5 severe problems 80% problems take approximately 15 days.

Here are some graphs and table that shows response time and severity of problems:



1. **Maintenance Process and Timeline**

Swimlane diagram:

****

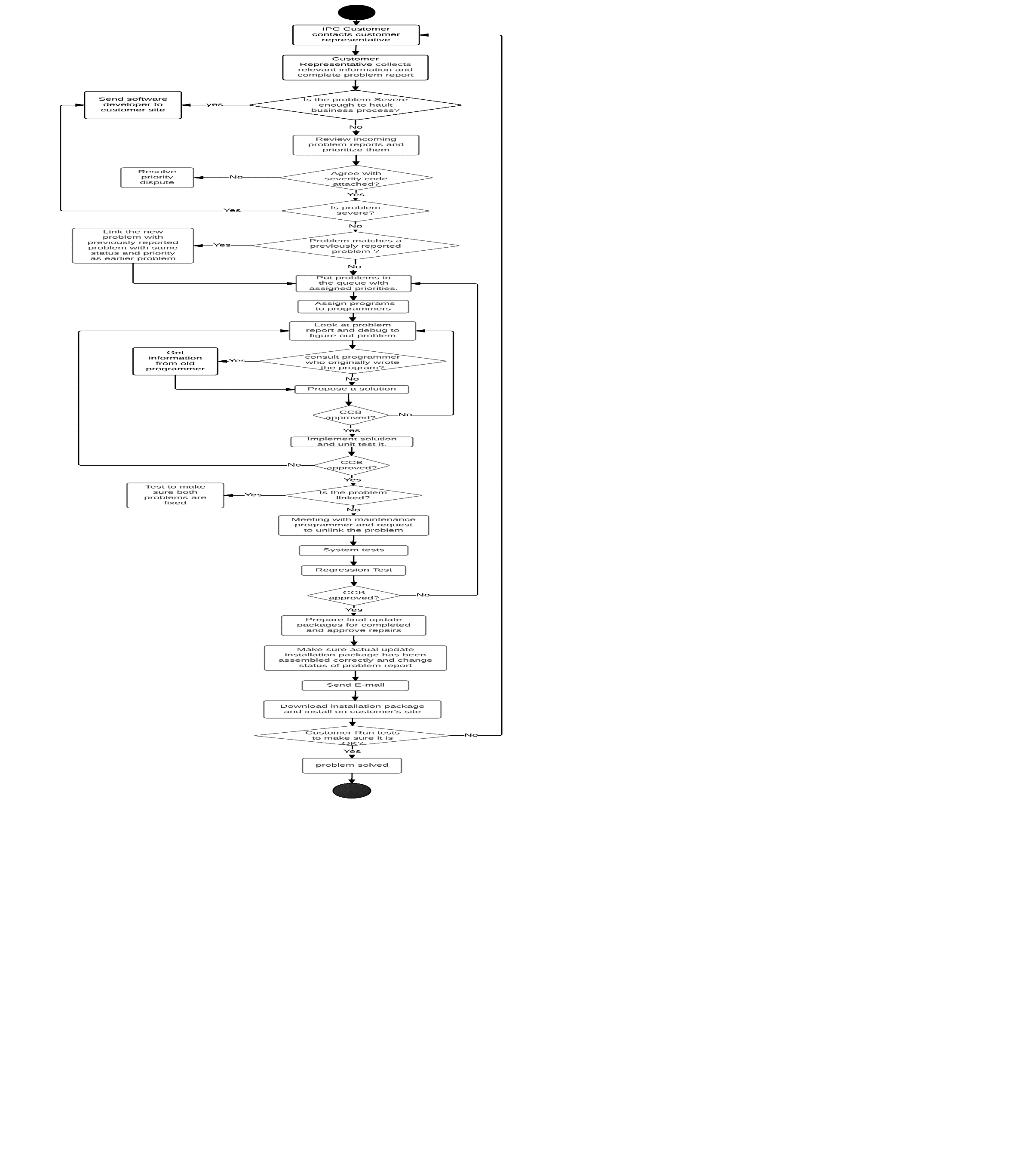
(PDF file of swim Lane diagram attached with name “A4 CSE6329 2015fa Report diagram kathuria tasmeen “)

Organizations involved and role they play in the process:

1. **IPC customer** contacts IPC’s customer representative in case of a software failure. Also, when he receives an update of the problem, he runs some tests to check if the results are ok.
2. **Customer Representative** collects the relevant information and completes a problem report. The report is sent to maintenance clerk. He then checks for the severity of the problem. If the problem is severe enough to halt a business process, he directly talks to the maintenance manager and she sends a software developer to customer site immediately. Customer representative also sends a weekly status update to the customer. Furthermore, once he gets the final repair, he downloads the installation package and installs it on the customer site.
3. **Maintenance clerk** reviews the incoming problem reports and prioritizes them. The customer representative usually assigns the severity code but sometimes he disagrees with them. In that case, he asks Maintenance manager what severity to assign. If he finds a severe problem he contacts Maintenance manager who in turn contacts Maintenance programming manager to send an engineer to the customer site. For other problems he puts them in the queue. Also, if he finds a problem that matches a previously reported problem he links them and put them to queue with the same status and priority. Later if Programmer thinks that the problems should not have been linked, he notifies maintenance clerk and he unlinks the problem and puts them back to the queue with the date they were unlinked on. Also, he ships the updates to the customer representative.
4. **Maintenance Manager Melinda shah** is one of the CCB members. Melinda is George Wilson’s supervisor and maintenance test staff also reports to her. She resolves priority disputes and also contacts Maintenance Programming manager regarding the disputes and in the case of severe problems. She also makes sure that actual update installation package has been assembled correctly.
5. **Maintenance Programming manager** is also one of the three members of CCB. She supervises the programmers and assigns them to work on a specific problem report. He is also responsible for handling priority disputes and sending an engineer to customer site in case of severe problems. He also changes the status of the problem report and prepare update installation package.
6. **Maintenance programmer** works for maintenance programming manager. He analyzes the problem report and debugs to figure out the problem. Since there is a lack of documentation he sometimes needs to consult the original programmer. Maintenance programmer then proposes a solution and gets it approved by CCB members. Once approved he implements the solution and unit test it. This implementation is again sent to CCB members for approval. If the programmer finds out that the problem is linked he tests to make sure that both the problems are linked and if he finds that problems are incorrectly linked he informs the maintenance clerk about this.
7. **Maintenance Test Team Lead** reports to Melinda Shah maintenance Manager. She tests all the repairs to see if they work. The main job is to run system-level tests after the problem has been repaired and unit tested by the programmer. They always have a meeting with maintenance programmer to explain how the program works and suggest some system test approaches. After the system tests are done they move towards the Regression tests and gets the results approved by CCB. Once the results are approved they prepare the final update packages.
8. **Quality Assurance manager** supervises QA staff and serves on both software development CCB and software maintenance CCB. He observes testing and programming. Also improves the coordination between development and maintenance organizations.
9. **Software Development manager** supervises software development for new products. He balances the programmer’s current job and severity of the maintenance problem.
10. **President and CEO** insist that software developers should meet new development product deadlines and give it more priority.

The organization chart for IPC

Flow chart for the process:



The processes that result in excess waiting in process are

* Long waiting queues caused by a company policy that one programmer should be assigned with only one problem at a time.
* Also according to company policy CCB members should meet once a week and all should be present for an approval. Since they are sometimes not available at the same time it leads to delays in the process as there is design waiting for approval to be coded.
* Regression tests are done once every few weeks and it is done only when multiple problems have been resolved. This causes significant delay.
* If a problem gets unlinked it goes to the queue with the date it was unlinked on rather than the date it was linked on.

Tasks that do not add value or constitute unnecessary rework are:

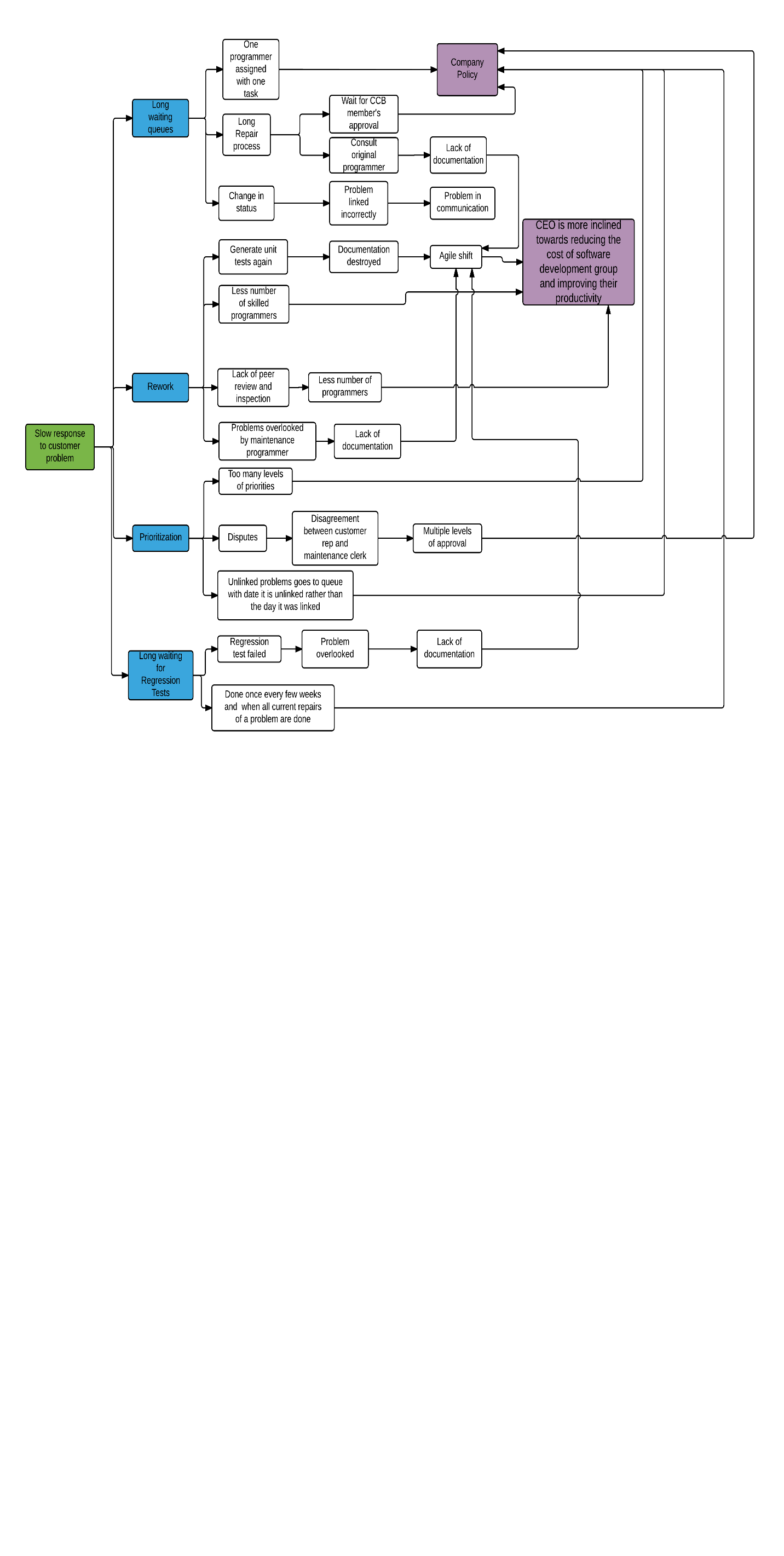
* Due to lack of documentation, maintenance programmers generate unit tests again.
* There are low skilled programmers in the maintenance organization.
* Lack of peer reviews and inspections
* Since there is no proper documentation available maintenance programmer sometimes overlook the problem.

Complex Task

* When a severe task is encountered at first step customer representative informs Maintenance manager and passes the report to maintenance clerk. Maintenance clerk again informs Maintenance manager for the task that is severe.

The significant causal factors are the old company policies and CEO being more inclined towards reducing the cost of software development team.

1. **Causal Model**

****

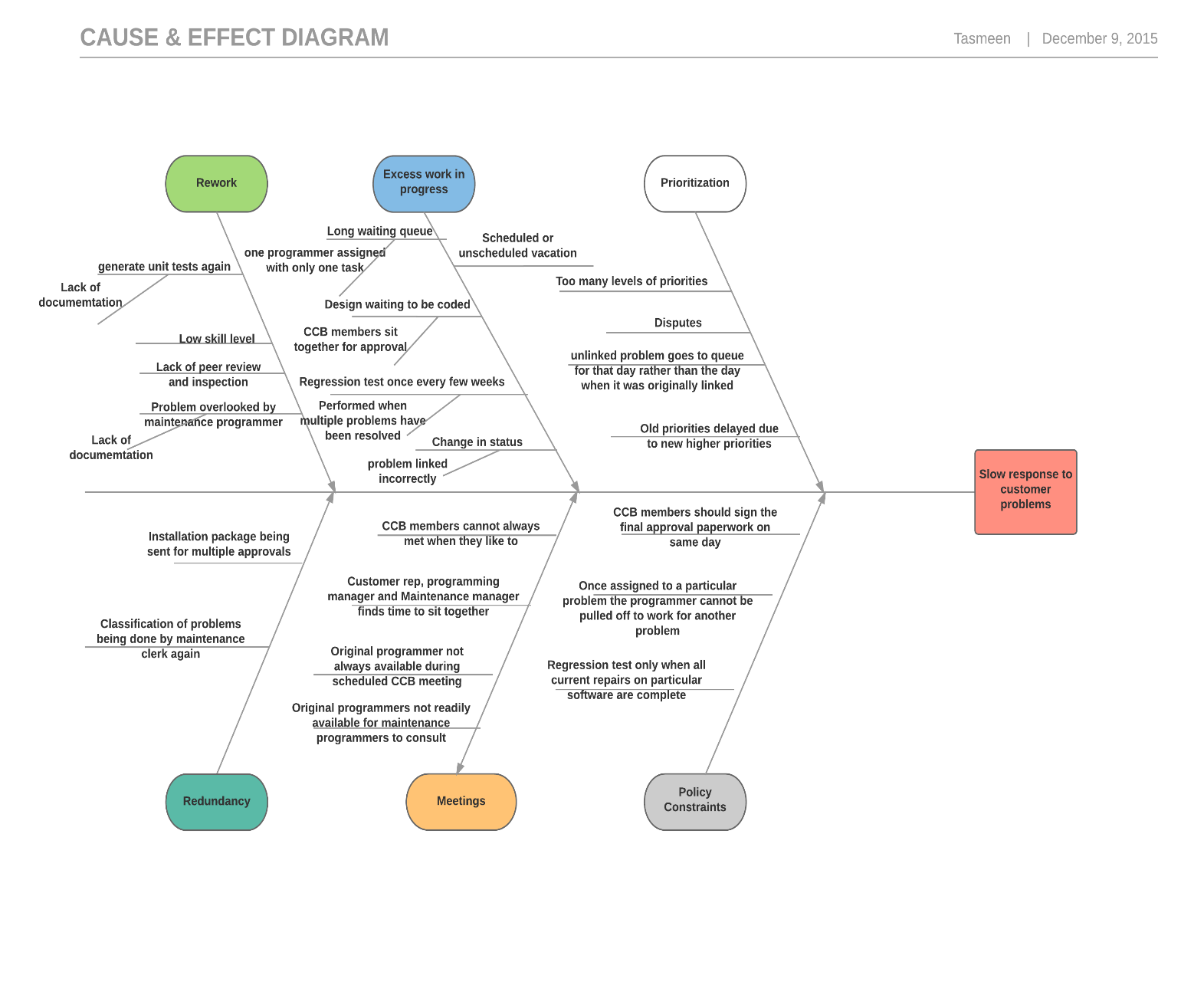
**Figure (a)**

The possible causes are:

* **Long waiting queues**: This is caused by a company policies - that one programmer should be assigned with only one problem at a time, Long repair process caused due to waiting for CCB approval or need to consult original programmer because of lack of documentation due to agile shift. Furthermore, CCB approval causes a delay because according to company policy they meet once a week and for final approval all the members should sign the paperwork on the same day. Long waiting queue is also caused by the change in status back to "waiting to be assigned" because of incorrect linking of problems.
* **Rework**: the main causes of Rework are
  + Generating unit tests again as due to agile shift the original programmers destroyed the unit tests then produced.
  + Lack of peer reviews and inspection
  + Problems overlooked by maintenance programmer due to lack of documentation caused by agile shift.
* **Priorities**: Due to company policy there are too many levels of priorities which cause inconsistencies. Also, as there are multiple levels of approvals it leads to disputes in deciding the priority levels. Moreover, when a problem is unlinked it goes to the queue with the date it is unlinked on rather than the date it was linked on.
* **Long waiting for regression test**: Since there is a lack of documentation due to agile shift, programmers sometimes overlook the problem which results in failure of regression test. Also, due to company policy the regression test is done once in a few weeks which leads to significant delays.

As we can observe in figure (a) there are 5 causes that are caused due to the company policies and 7 causes caused because CEO is more inclined towards improving the productivity of software development group and agile shift reduced their cost significantly. So, the later has a major impact on the problem.

1. **Root Cause Analysis**
   1. **Fishbone diagram**

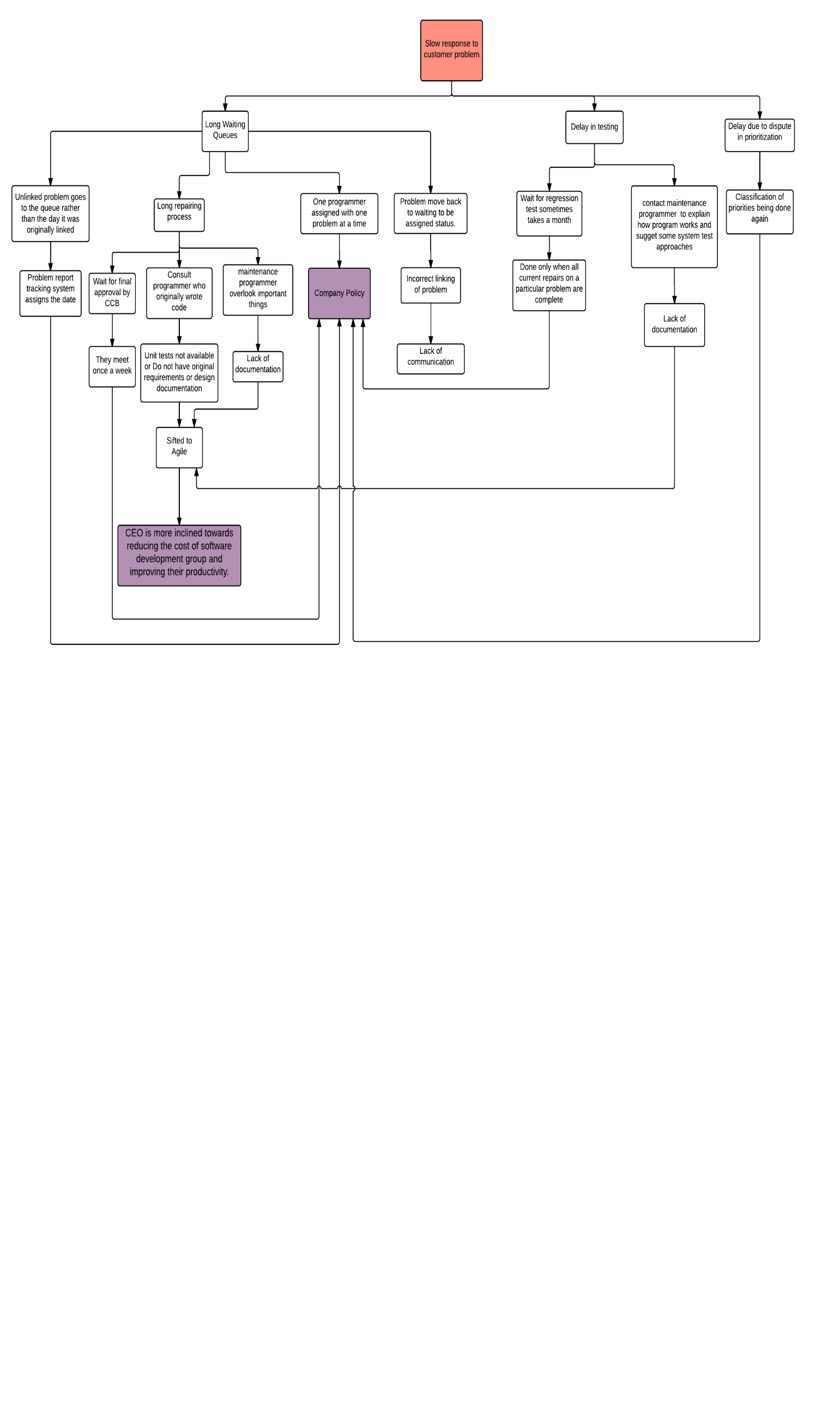
****

The fishbone diagram is useful for brainstorming possible causes of problems.

The categories of the problem discovered are Rework, Excess work in Progress, Prioritization, Redundancy, Communication, and Policy constraints.

The causes we found in Rework, Policy constraints and Excess work in progress are most significant. Following is the brief description of causes

* **Rework**: the causes of rework are generating unit tests again due to lack of documentation, low skilled programmers, lack of peer review and inspection, problem overlooked by maintenance programmer due to lack of documentation.
* **Excess work in progress**: the causes of excess work in progress are long waiting queues as one programmer is assigned with only one problem at a time, design waiting for approval as CCB members meet once a week and all should be present for the approval, Regression test is done once in a few weeks and when multiple problems are solved, when a problem is unlinked status changes back to “waiting to be assigned” and since programmers can be assigned with only one problem at a time and cannot be taken off the problem scheduled and unscheduled vacations lead to delays.
* **Priorities**: There are too many levels of priorities. Also, since priorities are being assigned at multiple levels, it leads to disputes. Unlinked problems go to the queues with the date it is unlinked on. Furthermore old priorities are delayed due to new higher priorities.
* **Redundancy**: There is little redundancy that is installation packages being sent for multiple approvals and classification of priorities being done again
* **Meetings**: CCB members are not able to meet when required that leads to delays, original programmers are not always available when required as they have pressure to meet new product deadlines.
* **Policy constraints**: The policy constraints imposed by are – CCB members should sign the paperwork on same working day but they meet once a week and all should be present, Once assigned to a problem the programmer cannot be pulled off to work for another problem and neither he is allowed to multitask, Regression tests are done once in every few weeks and only when multiple problems are solved.
  1. **Five whys**

****

**Figure (b)**

In five why’s we keep on asking “Why?” at each step until we find a root cause.

For the question to why there is a slow response to customer problems, we found 3 answers i.e. long waiting queues, delay in testing phase and delay due to dispute in prioritization.

For the question, why there are long waiting queues we further found 4 answers.

* Unlinked problems go to the queue with the date it is unlinked on rather than the date it was first linked on. The answer to it's why is Problem report tracking system assigns this date. For it's why – it is a company policy.
* Long repairing process because of 3 reasons. One, wait for final approval by CCB because they meet once a week. Why? Because it is a company policy. Two, consult programmer who originally wrote code because unit tests and requirements were not available due to agile shift. Three, maintenance programmer overlook things due to lack of documentation, Why? Because of the agile shift.
* One programmer is assigned with only one problem at a time. Why? Because it is a company policy.
* The problem sometimes moves back to "waiting to be assigned status" because there was incorrect linking of problem due to lack of communication.

For the question why there is delay in testing, there are two answers-

* Wait for regression tests sometimes takes a month because it is done only when multiple problems are solved. Why? It is a company policy.
* Contact maintenance programmers to generate unit tests due to lack of documentation. Why? Agile shift.

For question why there is a dispute in prioritization- classification of priorities are done again. Why? Company policy.

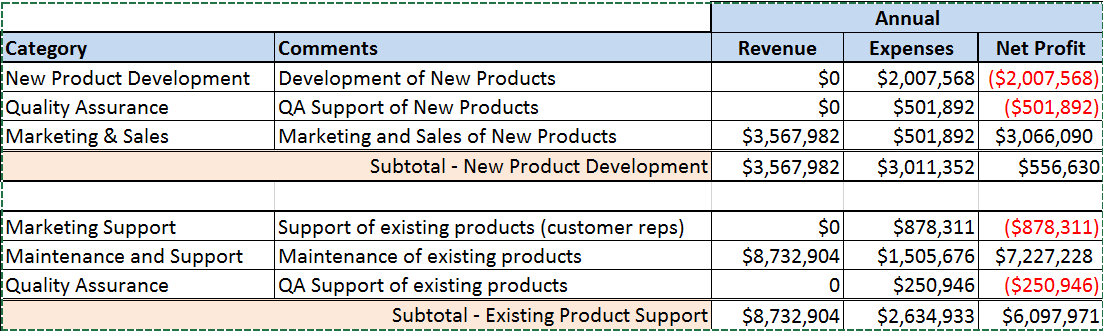
There was an agile shift because CEO is more inclines towards reducing the cost of software development group and improving their productivity.

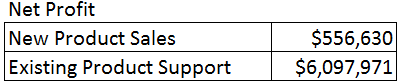
Here we found 2 main causes namely Company policies and CEO is more inclined towards reducing the cost of software development group and improving their productivity.

1. **Conclusions**

According to the analysis done, there are 2 root causes

1. “CEO is more inclined towards reducing the cost of software development group improving their productivity”. He seems to think IPC should make more profit on new software products. But according to IPC’s annual revenue data Maintenance organization makes more profit than new product development organization.





Due to Agile shift the software development programmers did not make requirements specifications document and also destroyed the unit tests produced. The lack of documentation has caused significant delays in maintenance process and is the major cause of the problem. This has led to delays because:

* Maintenance programmers need to consult original programmers and they are under pressure of meeting deadlines of new products.
* Generate unit tests again
* Less number of skilled programmers in maintenance organization
* Problems overlooked by maintenance programmers that lead to failure or regression test.

1. “Company policies that were implemented long time ago”. These company policies were instituted according to the scenario in past. Some of such policies are:

* CCB members should sit together for the approval was imposed so that all three members could discuss an issue at the same time if any.
* Regression tests are done once in every few weeks and when all the problem repairs are collected.
* A programmer is assigned with one problem at a time.
* Unlinked problems go to the queue with the date it is unlinked on rather than the date it was first linked.

1. **Recommendations**

Following are the recommendations:

* The biggest problem encountered is the lack of documentation. I suggest that software developers should not destroy unit tests and archive them. Also, in spite of agile shift requirements specifications document should be prepared so that maintenance programmers can refer them and the delays can be avoided.
* As we can see from financials, company’s major profit is from maintenance organization. There should be a few more programmers in the maintenance organization so that it would be easier for them to do peer reviews and inspections. Also, more number of skilled programmers should be part of the maintenance organization.
* Programmers are assigned with one problem at a time. This is not an efficient approach and leads to significant delays. Maintenance programmers are capable of multi-tasking and can handle it. So this restriction should be taken off.
* Programmers cannot be pulled off from the problems assigned to them leads to delays and sometimes inefficiency. If required, managers should be allowed to pull the programmers off the problems they are assigned to.
* CCB members meet once a week and all need to be present for final approval. This restriction was implicated when face to face meetings were considered important. But today with the advancement in technology services like skype, emails, smartphones etc. should be used and CCB members should meet more frequently as required.
* The number of priority tasks should be restricted by removing the priority from a task while assigning it to another one. Also, distribution of priorities should be monitored throughout the process.
* The tasks should be removed from the priority list when they don’t really need priority anymore.
* If customer representative is informing maintenance manager regarding the severe task, maintenance clerk should not inform maintenance manager for the same task again. The classification of priorities should be done at only one level.
* The unlinked task should go to the priority queue with the date it was first linked on and not the date on which it is unlinked.
* Regression tests are run once every few weeks. The policy was implemented when regression tests would take a whole week. But computers are now a lot faster. So, regression tests can be run more frequently to avoid significant delays.

The most important of all is generating documentation and not destroying it. I believe with above recommendations Acme Corporation will be able to improve the response time to customer problems.