[cover sheet 1 of 2]

2

ASSIGNMENT 2

**DEFECT ANALYSIS REPORT**

CSE 6329 -- SOFTWARE MEASUREMENT AND QUALITY ENGINEERING

Professor Dennis J. Frailey

**Fall, 2019**

NAME \_\_\_**Pavan Kumar Harikrishnan**\_\_

ID Number\_\_\_**1001641830**\_\_\_

NAME \_\_\_**Vismaya Menon**\_

ID Number\_\_\_**1001740500**\_

Grader Comments:

|  |  |  |
| --- | --- | --- |
| **Grading Template (student should not write on this page)** | | |
| **Section 1 – Introduction and Overview** | | |
| \_\_\_\_\_ (/9) | 1.1 \_\_\_\_\_ Purpose of This Report (1)  1.2 \_\_\_\_\_ Structure of Analysis Tool (4)  1.3 \_\_\_\_ Data Collection Overview (2)  1.4 \_\_\_\_\_ Summary of Analyses and Graphs (2) | (9 points) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Section 2 – Measures, Graphs and Analysis** | | | |
| **Section 2.1 – Post Release Quality** | | | |
|  | i (3 points)  **Overview**  (Purpose / Question / Definition / Collection) | ii (3 points)  **Graph** | iii **Analysis** & Discussion (6 points) |
| 2.1.a \_\_\_\_\_/12 |  |  |  |
| 2.1.b \_\_\_\_\_/12 |  |  |  |
| 2.1.c \_\_\_\_\_/12 |  |  |  |
| 2.1.d \_\_\_\_\_/12 |  |  |  |
| **Section 2.2 – Current Quality** | | | |
|  | i (3 points)  **Overview**  (Purpose / Question / Definition / Collection) | ii (3 points)  **Graph** | iii **Analysis** & Discussion (6 points) |
| 2.2.a \_\_\_\_\_/12 |  |  |  |
| 2.2.b \_\_\_\_\_/12 |  |  |  |
| 2.2.c \_\_\_\_\_/12 |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Section 3 – Summary and Recommendations** | | |
| 3.1 \_\_\_\_\_\_/3 | Summary |  |
| 3.2 \_\_\_\_\_\_/4 | Recommendations |  |

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_/100 | 🡸 **Total Assignment Grade** |

**Defect Analysis Report**

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

This report shows the results of analyzing three years of defect data on our released products. The purpose is to gain a greater understanding of the quality levels of our released products and to determine whether there is any correlation between software quality and other factors such as the programming language used, the development process used, or the time when the product was developed.

* 1. **Structure of Analysis Tool**

In order to analyze defect data, we have created a workbook of the following sheets. We have given the details of each sheet below.

1. A4data:

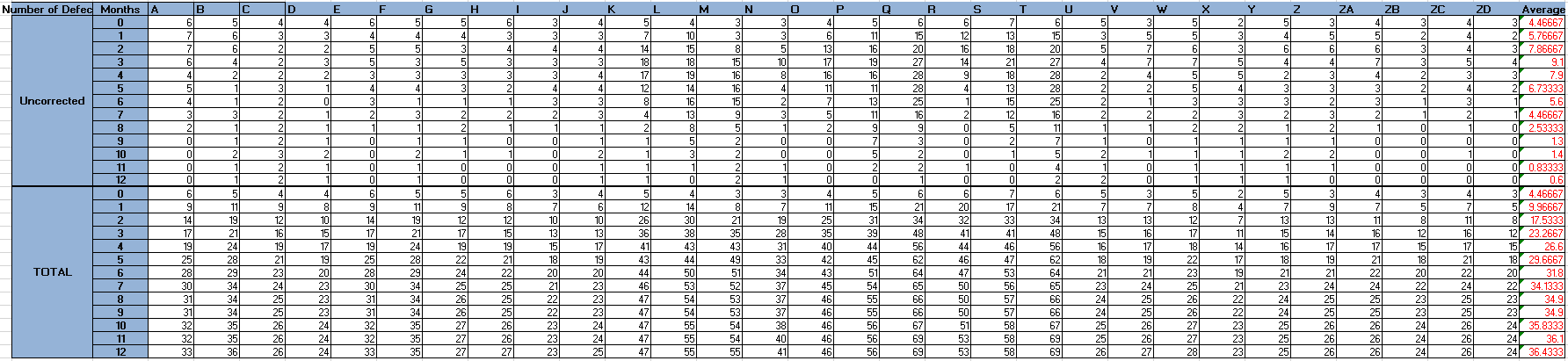
• This is the data that is given for analysis.

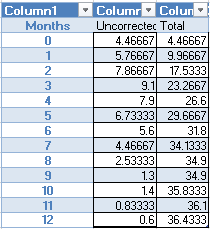
• It has precise information of their products, release month, new and corrected defects and the active period.

2. PRQ Average:

• This sheet has a table that shows the uncorrected and total defects. The total defects are the sum of new and corrected defects.

•This is calculated for each and every product for their active period and their average is calculated.

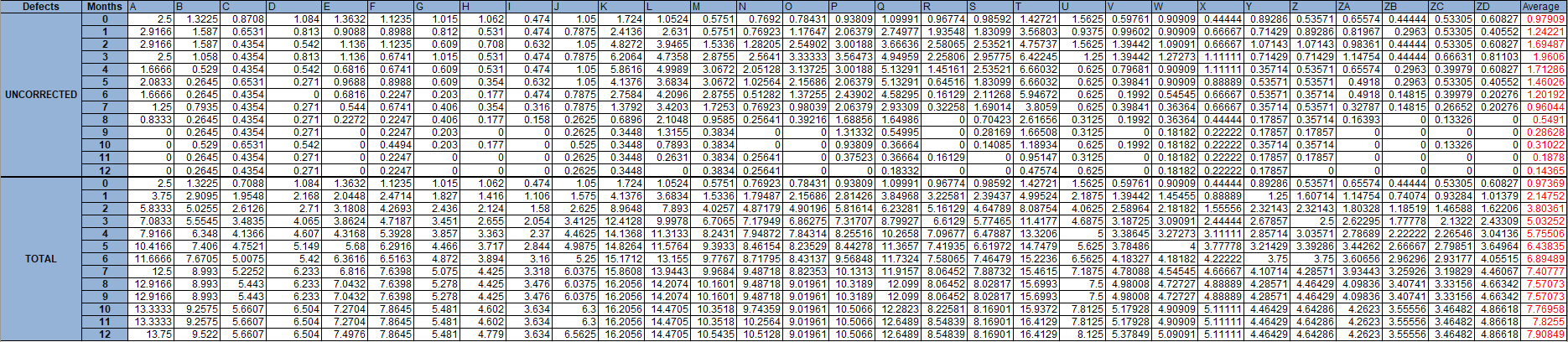


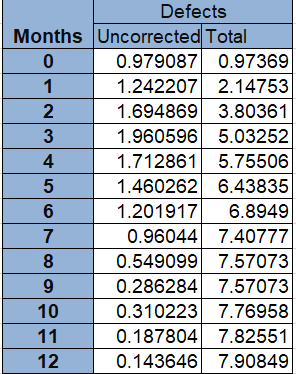


3. PRQ by Size:

• Here the table shows of calculated values of defects normalized to the size for all the products.

• The Average of the total and uncorrected defects is used to plot the graph.





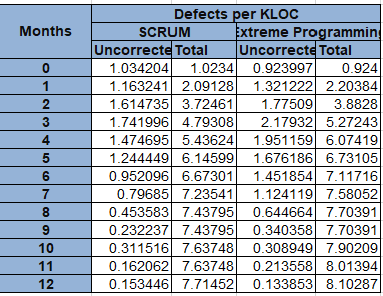
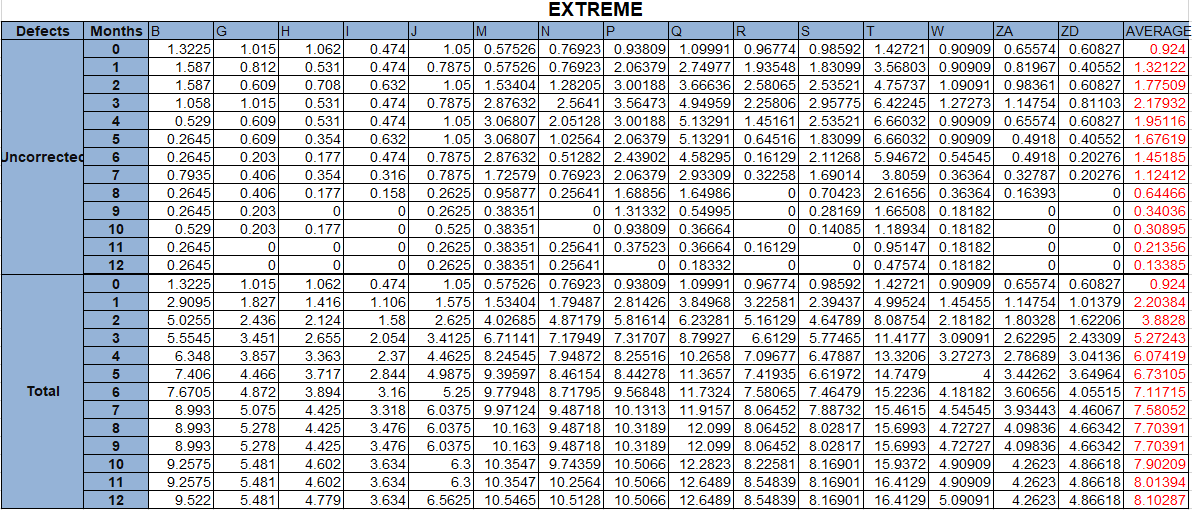
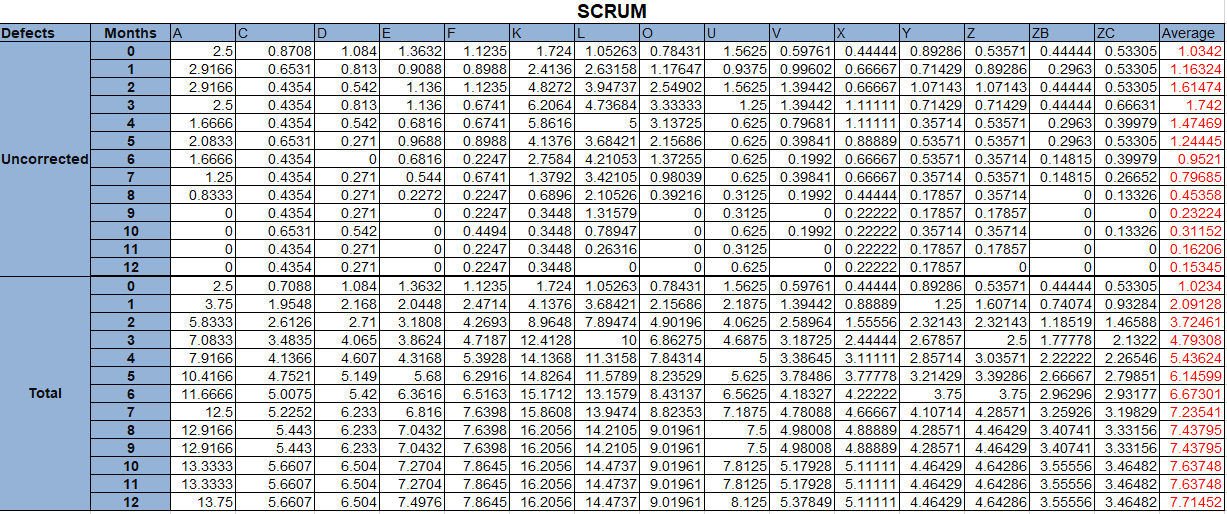
4. PRQ by process:

• This has 3 tables, based on methodology: Scrum and Extreme and the table of defects per 1000 lines of code.

• First table represents total defects and total uncorrected defects normalized by size for products developed using Extreme Programming method.

• Second table represents the total defects and total uncorrected defects normalized by size for products developed using SCRUM method.

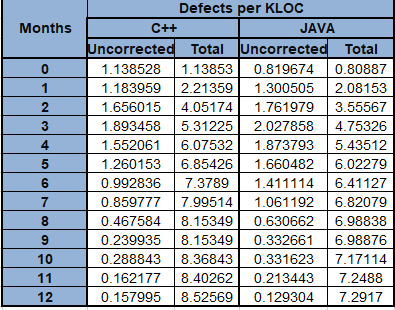
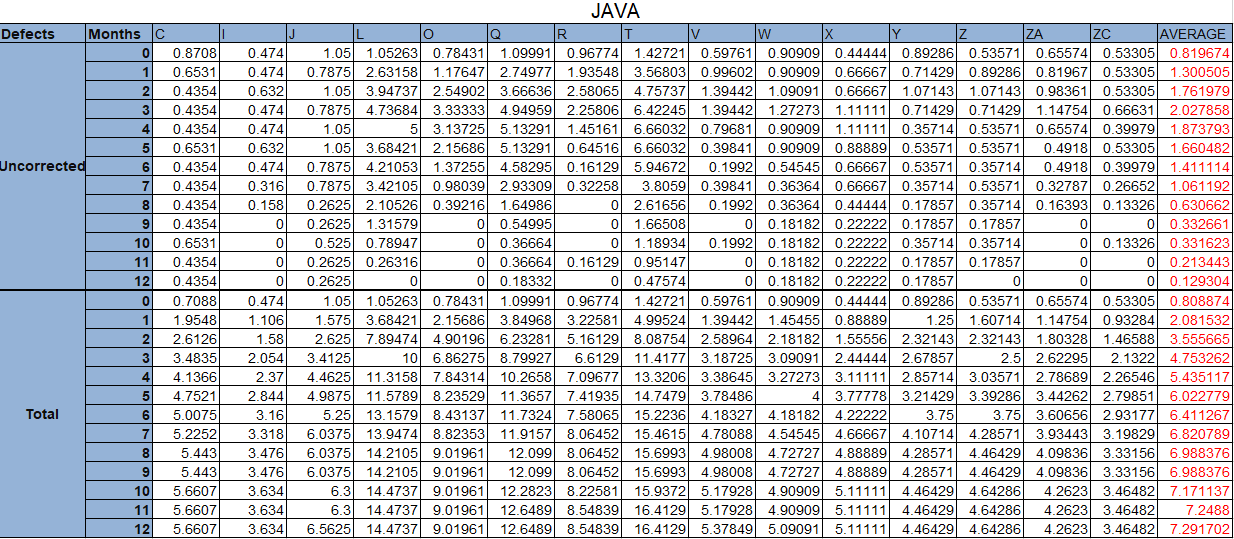
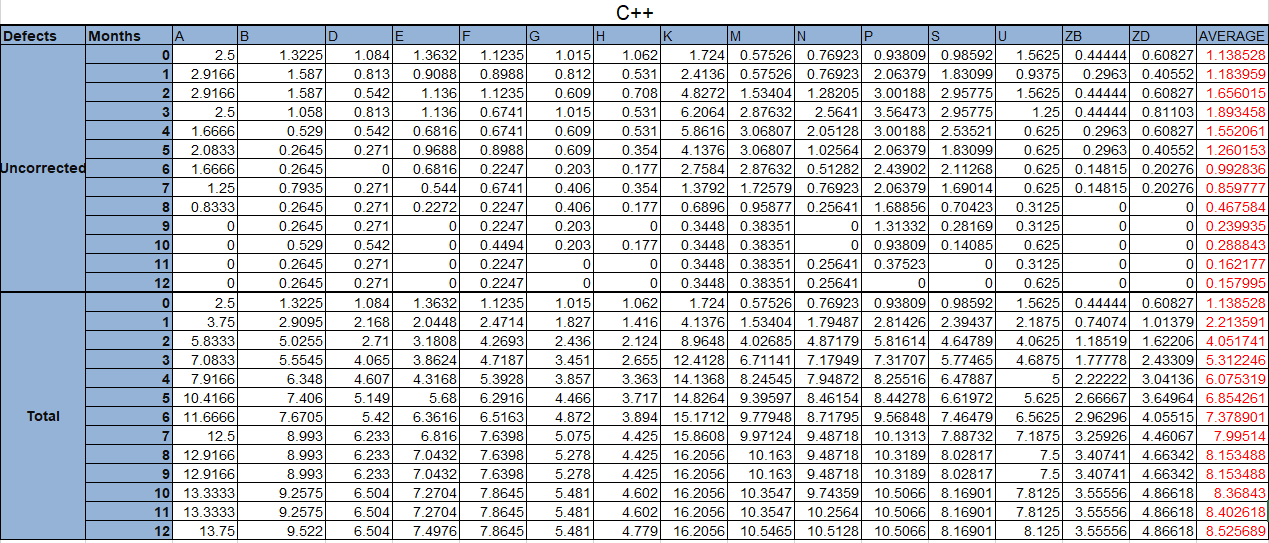
• Third table represents the average of total defects and total uncorrected defects for these two methods.



5. PRQ by language:

• This sheet has two tables showing the total defects and total uncorrected defects normalized by size for products developed using C++ and Java programming language.

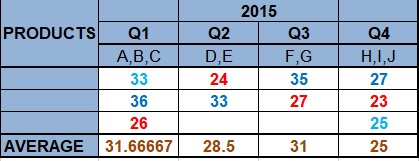
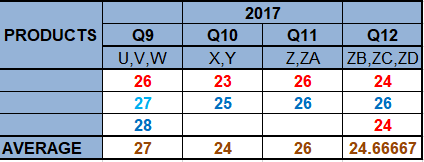
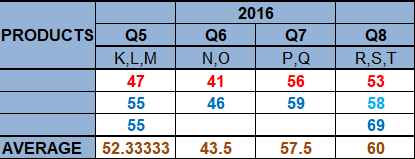
• The average of these two programming languages is computed and plotted.

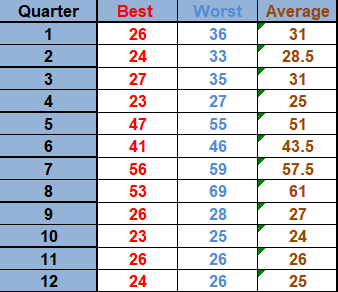


6. PRQ History:

• There are 3 tables giving the averages for the products over a span of 3 years i.e. 2015,2016 and 2017.

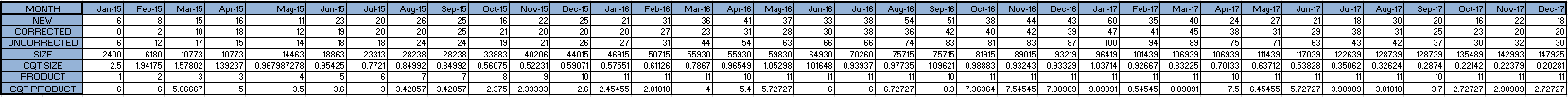
• The last table computes the Best case, Worst case and the Average case of the number of defects of the products quarterly.





7. CQT Data:

This table is used as reference to make the current quality total and plotting in the further sheets:



8. Current Quality Total:

• This chart shows current quality for each month for 2015, 2016 and 2017 with all active products for that particular month.



9. Current Quality Total normalized by Size:

• This shows the values of total number of uncorrected defects each month for all active products over a span of 3 years that is normalized by the size of the software product.

+

10. Current Quality Total normalized by number of products:

• This shows the values of total number of uncorrected defects each month for all active products over a span of 3 years that is normalized by the products that are active.



11. Variables:

|  |  |
| --- | --- |
| This sheet describes the variables used in the work book:  **Variables that Might Affect Results** | |
| **Environment** | |
| **People (programmer skill level)** | |
| **People (domain knowledge)** | |
| **x** | **Language** |
| **x** | **Development Process** |
| **Application Domain** | |
| **Development tools** | |
| **Management style** | |
| **Priorities (Pressure to meet deadlines vs quality)** | |

* 1. **Data Collection Overview**
* Data required to perform Defect analysis has been collected from the years 2015 to 2017 for all the 30 software products.
* This data is stored in excel spreadsheet named a2data.xlsx.Extra sheets have been added to that workbook to show our calculations and graphs
* Defect data for 30 products includes corrected defects, new defects and total defects for the active period of the product (12 months).
* This has precise data about the products that like programming language used for a particular product (Java and C++), development process used (SCRUM or Extreme programming) and thier size, for each product.

**1.4 Summary of Analyses and Graphs**

|  |  |  |
| --- | --- | --- |
| **Reference** | **Analyses** | **Description** |
| 2.1 | Post Release Quality | Average of all uncorrected defects and total defects |
| 2.1.a | Post Release Quality- Normalized by Size | Average of all uncorrected defects and total defects – normalized by size |
| 2.1.b | Post Release Quality- Normalized by Process | Average of all uncorrected defects and total defects – normalized by process |
| 2.1.c | Post Release Quality- Normalized by Programming language | Average of all uncorrected defects and total defects – normalized by language |
| 2.1.d | Post Release Quality- History | Average of all uncorrected defects and total defects – taken for each year, showing best, average and worst conditions. |
| 2.2.a | Current Quality Total | The total uncorrected defects for all active products |
| 2.2.b | Current Quality Total – Normalized by size | The total uncorrected defects for all active products- Normalized by size of the product |
| 2.2. | Current Quality Total – Normalized by active product | The total uncorrected defects for all active products- Normalized by active product |

1. **Measures, Graphs and Analysis**
   1. **Post Release Quality**

**2.1.a. Post-Release Quality Average, Normalized by Size**

**i. Overview**

* Post Release Quality describes the defects that are detected after the product has been released. These defects surface when the user begins to use the product. Analysis of such defects will be done for data that has been collected over a long period of time.
* Some graphs require information like Release date, Products list, development process and language, size of the code for each product, defect count. (these are the topics on which we are doing analysis in this project as per the requirements) In every graph we try to present how many defects are left uncorrected, how the normalization of defects makes a difference, what impact the various development languages and development processes have on the number of defects and based on the number of lines of code say 1000 lines of code here, how the defects are being analyzed for each of the 3 years for all products through A to ZD.
* We are using line graphs to represent theour analysis.

**ii. Graph**

The graph below shows PQR Average normalized by size.



**iii. Analysis and Discussion**

1. **General Discussion**

* This graph is normalized by size. This helps us understand if defect levels are due to size.
* Based on this graph, the total defects are increasing steadily.
* As the defect rate is showing a stable decrease from the 5th month of it’s release.
* Around the 10th month of release, the uncorrected defects are very less.

1. What you learned about the various defect/failure rate hypotheses

| Hypothesis | What this graph tells us about this hypothesis. | Summary (-- - 0 + ++) |
| --- | --- | --- |
| 1. There are not any significant failure rate increases. | We can see an increase in uncorrected defects rate from 1st month to 4th month, and in that period the total defects are also gradually increasing.  Though there is an increase in total defects after the 5th month, the uncorrected defects are decreasing. This supports the hypothesis. | - |
| 1. Higher defect levels are due to programming language. | In this graph, there is two lines (Uncorrected Defects and Total Defects) So it is not possible to judge whether higher defects are due to programming language. | 0 |
| 1. Higher defect levels are due to development process. | In this graph, there is two lines (Uncorrected Defects and Total Defects) So it is not possible to judge whether higher defects are due to development process. | 0 |
| 1. Defect increases are due to the fact that new products are larger. Defects/LOC are the same, | As the number of months increases, the total defects increases. From this we can see that as new products are larger, the total defects are also increasing. This graph supports the hypothesis as it is normalized by size. | ++ |
| 1. Increased defects are due to the fact that we have more products in use. | As we move across the horizontal axis, we can see the number of defects increasing. This is due to more number of products. Thus the graph supports the hypothesis | + |
| 1. The quality assurance program in 2017 improved things. | This graph tells us about the PQT by size for 12 months. It doesn’t talk about Quality Assurance program. | 0 |

**2.1.b Post Release Quality Average for All Products, by Development Process**

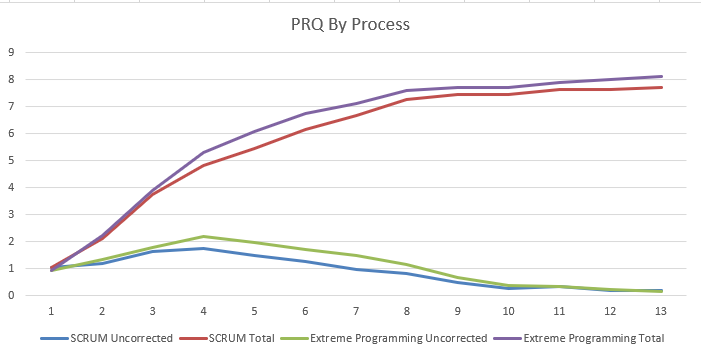
**i. Overview**

This graph helps to distinguish the effect of development process – Extreme Programming and SCRUM on the total defects and uncorrected defects

**ii. Graph**

The graph below shows Post Release Quality Average by development process –

SCRUM and Extreme programming for all products.



**iii. Analysis and Discussion**

1. **General Discussion**

* This graph is normalized by size and has four lines – SCRUM Total, SCRUM Uncorrected, Extreme Total, Extreme Uncorrected.
* Based on this graph, the total defects for Extreme programming is higher when compared to the Total defects of Scrum. Also the Uncorrected Defects for Extreme is more than the Uncorrected Defects of the Scrum.
* Towards the end there is no much difference in the Uncorrected errors for Extreme and SCRUM.

1. What you learned about the various defect/failure rate hypotheses

| Hypothesis | What this graph tells us about this hypothesis. | Summary (-- - 0 + ++) |
| --- | --- | --- |
| 1. There are not any significant failure rate increases. | We can see a steep increase in failure rate from 1st month to 4th month in both SCRUM and Extreme Programming. As the Number of Total defects and Uncorrected Defects increases simultaneously for both Extreme and SCRUM. So, the graph refutes the hypothesis, as there is a significant increase in the failure rate. | - |
| 1. Higher defect levels are due to programming language. | In this graph, we have four lines (Uncorrected, Total) for Extreme and Scrum. It doesn’t explain about programming language. | 0 |
| 1. Higher defect levels are due to development process. | In this graph, the total defects for Extreme programming is higher when compared to the Total defects of Scrum. Also the Uncorrected Defects for Extreme is more than the Uncorrected Defects of the Scrum. Thus the average values of both are the same. This graph doesn’t support the hypothesis | -- |
| 1. Defect increases are due to the fact that new products are larger. Defects/LOC are the same, | In this graph, Total Defects for both SCRUM and Extreme increases as we go across the x axis from left to right. This doesn’t talk about the size of the products, and thus doesn’t talk about the hypothesis. | 0 |
| 1. Increased defects are due to the fact that we have more products in use. | This doesn’t talk about the number of the products, and thus doesn’t talk about the hypothesis. | 0 |
| 1. The quality assurance program in 2017 improved things. | This doesn’t talk about the quality assurance program, and thus doesn’t talk about the hypothesis. | 0 |

**2.1.c Post Release Quality Average for All Products, by Programming Language**

**i. Overview**

This graph helps to distinguish the effect of Programming Language – C ++ and Java on the total defects and uncorrected defects

**ii. Graph**

The graph below shows Post Release Quality Average by development process –

SCRUM and Extreme programming for all products.

**iii. Analysis and Discussion**

1. **General Discussion**

* This graph is normalized by size and has four lines – C ++ Total, C ++ Uncorrected, Java Total, Java Uncorrected.
* Based on this graph, the total defects for C++ is higher when compared to the Total defects of Java. But the Uncorrected Defects for C++ is less than the Uncorrected Defects of the Java.
* Towards the end there is not much difference in the Uncorrected errors for C++ and Java.

1. What you learned about the various defect/failure rate hypotheses

| Hypothesis | What this graph tells us about this hypothesis. | Summary (-- - 0 + ++) |
| --- | --- | --- |
| 1. There are not any significant failure rate increases. | We can see a steep increase in failure rate from 1st month to 4th month in both Java and C++. As the Number of Total defects and Uncorrected Defects increases simultaneously for both Java and C++. So, the graph refutes the hypothesis, as there is a significant increase in the failure rate. | - |
| 1. Higher defect levels are due to programming language. | In this graph, the total defects are more for C++ than Java. Also the uncorrected defects are less for C++ than Java. So C++ is easier to be corrected and maintained compared to Java. | ++ |
| 1. Higher defect levels are due to development process. | In this graph, we have four lines (Uncorrected, Total) for C++ and Uncorrected, Total for Java. It doesn’t explain about development Process. | 0 |
| 1. Defect increases are due to the fact that new products are larger. Defects/LOC are the same, | This doesn’t talk about the size of the products, and thus doesn’t talk about the hypothesis. | 0 |
| 1. Increased defects are due to the fact that we have more products in use. | This doesn’t talk about the number of the products, and thus doesn’t talk about the hypothesis. | 0 |
| 1. The quality assurance program in 2017 improved things. | This doesn’t talk about Quality Assurance program, and thus doesn’t talk about the hypothesis. | 0 |

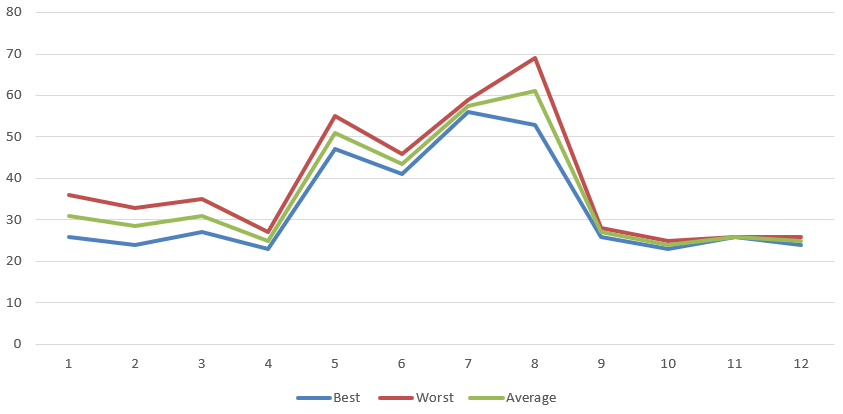
**2.1.d Post Release Quality History**

**i. Overview**

 This graph helps to understand the quality of the products shipped over several years in terms of defect rates. And it explains the change in scale over a period.

**ii. Graph**

Post Quality History Graph gives a picture on number of know defects are there in all currently active Products.



**iii. Analysis and Discussion**

1. **General Discussion**

* This graph is normalized by size and has three lines – best (least number of defects), worst (highest number of defects) and average.
* Based on this graph, the best, worst and average increases drastically after 4th Quarter and remains high till 8th quarter with a light decrease in between. And decreases drastically after the 8th Quarter.
* All three lines is steady and low after the 9th Quarter.

1. What you learned about the various defect/failure rate hypotheses

* -- The graph strongly refutes the hypothesis
* - The graph refutes the hypothesis but not strongly
* 0 The graph says little or nothing about the hypothesis
* + The graph supports the hypothesis but not strongly
* ++ The graph supports the hypothesis strongly

| Hypothesis | What this graph tells us about this hypothesis. | Summary (-- - 0 + ++) |
| --- | --- | --- |
| 1. There are not any significant failure rate increases. | In this graph, all the three lines (best, worst, average) increases after the 4th Quarter; this shows there is a increase in failure rate. So, this graph refutes the hypothesis. | -- |
| 1. Higher defect levels are due to programming language. | In this graph, it considers all products irrespective of the programming language. So, it doesn’t explain the hypothesis. | 0 |
| 1. Higher defect levels are due to development process. | In this graph, it considers all products irrespective of the development Process. So, it doesn’t explain the hypothesis. | 0 |
| 1. Defect increases are due to the fact that new products are larger. Defects/LOC are the same, | In this graph, all three lines increases after the 4th quarter. This might be because of the larger new products. So this graph supports, the hypothesis. | + |
| 1. Increased defects are due to the fact that we have more products in use. | In this graph, every quarter has no more than three products. So, this graph doesn’t talk about the hypothesis. | 0 |
| 1. The quality assurance program in 2017 improved things. | In this graph, all three lines are flat and lies very low after the 9th quarter which is in 2017. So, this graph supports the hypothesis. | ++ |

* 1. **Current Quality**

**2.2.a Current Quality Total**

**i. Overview**

Graph current quality Total gives idea on defects in all active products for 3 year by each month. The vertical axis will be “total defects”.

**ii. Graph**



**iii. Analysis and Discussion**

1. **General Discussion**

* In CQT graph, there is one line which shows the defects over months for three years.
* The defects are steady from Jan- 15 till Sep-15 and gradually increases from Oct-15. Again, the defects decrease gradually from Jan-17.

1. What you learned about the various defect/failure rate hypotheses

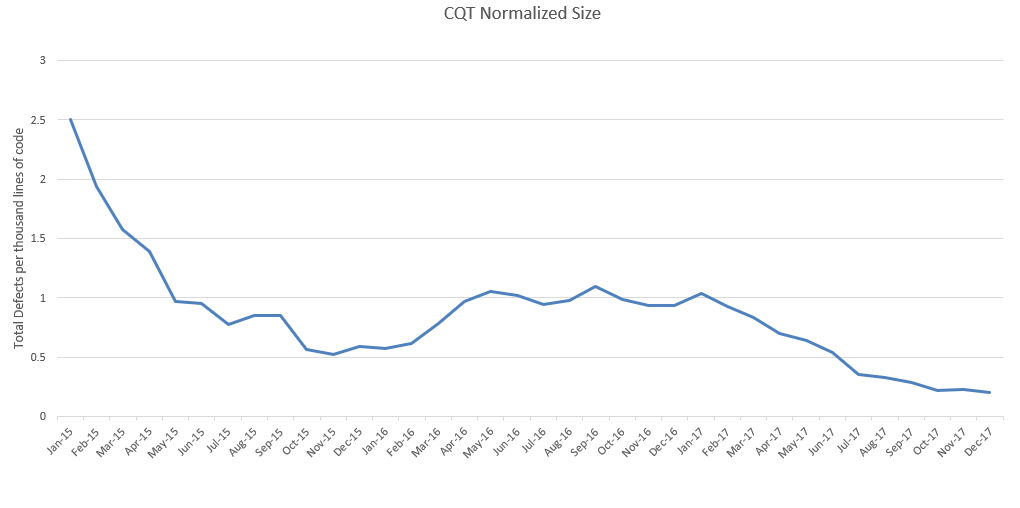
| Hypothesis | What this graph tells us about this hypothesis. | Summary (-- - 0 + ++) |
| --- | --- | --- |
| 1. There are not any significant failure rate increases. | In this graph, there is a gradual increase in defects from Oct-15 which shows there is an increase in failure rate. So, the graph refutes the hypothesis. | -- |
| 1. Higher defect levels are due to programming language. | In this graph, it considers all active products irrespective of the programming language. So, it doesn’t explain the hypothesis. | 0 |
| 1. Higher defect levels are due to development process. | In this graph, it considers all active products irrespective of the development Process. So, it doesn’t explain the hypothesis. | 0 |
| 1. Defect increases are due to the fact that new products are larger. Defects/LOC are the same, | In this graph, the defects increase from Oct-15 this might be because of larger new products. So, this graph supports the hypothesis. | + |
| 1. Increased defects are due to the fact that we have more products in use. | In this graph, the defects increase from Oct-15 this might be because there are more products. So, this graph supports the hypothesis. | + |
| 1. The quality assurance program in 2017 improved things. | In this graph, the defects decrease from Jan-17, this can be due to the assurance program. So, it strongly supports the hypothesis. | ++ |

**2.2.a Current Quality Total Normalized by Size**

**i. Overview**

In this graph, each value is normalized by the size of the corresponding software product. The vertical axis will be “total defects per thousand lines of code”.

**ii. Graph**



**iii. Analysis and Discussion**

1. **General Discussion**

* In CQT graph, there is one line which shows the Total defects per thousand lines of code over months for three years.
* The normalized defects are high in Jan-15 till Nov-15. From Dec-15 there is a gradual increase with fluctuations till Jan-17. After Jan-17 there is a decrease in the defects.

1. What you learned about the various defect/failure rate hypotheses

| Hypothesis | What this graph tells us about this hypothesis. | Summary (-- - 0 + ++) |
| --- | --- | --- |
| 1. There are not any significant failure rate increases. | This graphs shows a high defect rate in Jan-15. There is a gradual decrease from then on. From Jan-16 to Jan-17 there is again a slight increase in defect rate, but it is much lesser than that of Jan-15 time period. The defects are decreasing gradually from Feb-17, thus saying there is not any significant failure rate | + |
| 1. Higher defect levels are due to programming language. | In this graph, it considers normalized defects based on size irrespective of the programming language. So, it doesn’t explain the hypothesis. | 0 |
| 1. Higher defect levels are due to development process. | In this graph, it considers normalized defects based on size irrespective of the development Process. So, it doesn’t explain the hypothesis. | 0 |
| 1. Defect increases are due to the fact that new products are larger. Defects/LOC are the same, | In this graph, Jan-15 has the highest defect though the KLOC is less. There is a gradual decrease in defects even though the KLOC is gradually increasing with time. So, this graph doesn’t support the hypothesis. | - |
| 1. Increased defects are due to the fact that we have more products in use. | In this graph, it considers normalized defects based on size irrespective of number of active products. So, it doesn’t explain the hypothesis. | + |
| 1. The quality assurance program in 2017 improved things. | In this graph, the normalized defects decrease from Jan-17, this can be due to the assurance program. So, it supports the hypothesis. | ++ |

**2.2.c Current Quality Total Normalized by Number of Products**

**i. Overview**

In this graph, each value is normalized by the total number of currently active products each month.

**ii. Graph**



**iii. Analysis and Discussion**

1. **General Discussion**

* In CQT graph, there is one line which shows the Total defects per thousand lines of code normalized by total number of active products for three years.
* The normalized defects are high and decreases gradually till Nov-15. After which there is a steep increase till Jan-17. Then, we can see a gradual decrease after Feb-17.

1. What you learned about the various defect/failure rate hypotheses

| Hypothesis | What this graph tells us about this hypothesis. | Summary (-- - 0 + ++) |
| --- | --- | --- |
| 1. There are not any significant failure rate increases. | This graph starts with a increased defect rate in Jan-15. There is a gradual decrease from then on. From Jan-16 to Jan-17 the defects show a steep increase. There is a gradual decrease from after Feb-17. This graph shows a significant failure rate | - |
| 1. Higher defect levels are due to programming language. | In this graph, it considers normalized defects based on KLOC by active products irrespective of the programming language. So, it doesn’t explain the hypothesis. | 0 |
| 1. Higher defect levels are due to development process. | In this graph, it considers normalized defects based on KLOC by active productsirrespective of the development Process. So, it doesn’t explain the hypothesis. | 0 |
| 1. Defect increases are due to the fact that new products are larger. Defects/LOC are the same, | In this graph, it considers normalized defects based on KLOC by active products irrespective of the size. So, it doesn’t explain the hypothesis. | 0 |
| 1. Increased defects are due to the fact that we have more products in use. | In this graph, is Jan-15 has high defect though the active products are less. There is a gradual increase in defects from Jan-16, though the active products are higher. So, this graph strongly refutes the hypothesis. | -- |
| 1. The quality assurance program in 2017 improved things. | In this graph, the normalized defects decrease from Jan-17, this can be due to the assurance program. So, it supports the hypothesis. | ++ |

**3.0 Summary and Recommendations**

**3.1 Summary**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hypothesis | Graphs / Measures | | | | | | | Overall | Comments |
| 1a | 1b | 1c | 1d | 2a | 2b | 2c |
| **1 no issue** | - | - | - | -- | -- | + | -- | - | From all graphs we can conclude that there are issues (Significant increase in failure rates). |
| **2 language** | 0 | 0 | ++ | 0 | 0 | 0 | 0 | ++ | C++ is easier to debug and fix than Java, as we saw from the graph |
| **3 process** | 0 | -- | 0 | 0 | 0 | 0 | 0 | -- | The development process has nothing to do with the failure rate of the products. |
| **4 size** | ++ | 0 | 0 | + | + | - | 0 | + | The size of the product slightly does increase the failure rate. |
| **5 Number of Products** | + | 0 | 0 | 0 | + | + | -- | + | The greater number of products slightly does increase the failure rate. |
| **6 QA program** | 0 | 0 | 0 | ++ | ++ | ++ | ++ | ++ | The quality assurance program strongly decreases the defect rates. |

**3.2 Recommendations**

* Strongly recommend reinstating the Quality Assurance program, as seen from the graphs, there’s a significant decrease in failure rates due to the QA program.
* We would recommend using C++ over Java, as the developers seem more comfortable in debugging the defects in C++. Even though the graph shows Higher defect rates in C++, the uncorrected defect rates are lesser for C++.
* The failure rates due to the size and number of products will decrease with increased Testing and Quality Assurance.
* Development process plays no role in the failure rate, so the teams can choose either method for their product.