[cover sheet 1 of 2]

2

ASSIGNMENT 2

**DEFECT ANALYSIS REPORT**

CSE 6329 -- SOFTWARE MEASUREMENT AND QUALITY ENGINEERING

Professor Dennis J. Frailey

**Spring, 2017**

NAME \_\_\_**PATIL J., VISHWEKAR A.**\_\_\_

ID Number\_\_\_**1001228861**\_\_\_

[cover sheet 2 of 2]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Grading template Student do not write inside this box** | | | | | | | |
| \_\_\_\_\_ (/16) | 1.0 Description of Analysis Tool (spreadsheet)  1.1 \_\_\_\_ Purpose (1) 1.2 \_\_\_\_ Data Spreadsheet (3) 1.4 \_\_\_\_ Analysis Tool (12) | | | | | | (16 points) |
|  | 2.1 Details for each of the Post Release Quality graphs | | | | | | |
| \_\_\_\_\_\_\_ (/2) | 2.1.1 Overview of Post Release Quality (PRQ) graphs (2) | | | | | | |
|  | 2.1.2 … 2.1.7 Details of Individual Post Release Quality graphs (10 points each) | | | | | | |
| Introduction & Graph  (2.1.x), (2.1.x.1)  (2 points) | Analysis & Discussion  (2.1.x.2)  (2 points) | Procedure Used to Produce the Graph (2.1.x.3) | | | | |
| Base Metrics Collected  (2.1.x.3.1)  (1 point) | Compound Metrics  (2.1.x.3.2)  (1 point) | Data Refinement  (2.1.x.3.3)  (2 points) | How to Interpret  (2.1.x.3.4)  (2 points) | |
| Example  Product A  \_\_\_\_\_ (/10) | (2.1.2.1)  \_\_\_\_\_ | (2.1.2.2)  \_\_\_\_\_ | (2.1.2.3.1)  \_\_\_\_\_ | (2.1.2.3.2)  \_\_\_\_\_ | (2.1.2.3.3)  \_\_\_\_\_ | (2.1.2.3.4)  \_\_\_\_\_ | |
| Average  \_\_\_\_\_ (/10) | (2.1.3.1)  \_\_\_\_\_ | (2.1.3.2)  \_\_\_\_\_ | (2.1.3.3.1)  \_\_\_\_\_ | (2.1.3.3.2)  \_\_\_\_\_ | (2.1.3.3.3)  \_\_\_\_\_ | (2.1.3.3.4)  \_\_\_\_\_ | |
| Normalized by Size  \_\_\_\_\_ (/10) | (2.1.4.1)  \_\_\_\_\_ | (2.1.4.2)  \_\_\_\_\_ | (2.1.4.3.1)  \_\_\_\_\_ | (2.1.4.3.2)  \_\_\_\_\_ | (2.1.4.3.3)  \_\_\_\_\_ | (2.1.4.3.4)  \_\_\_\_\_ | |
| PRQ By  Process  \_\_\_\_\_ (/10) | (2.1.5.1)  \_\_\_\_\_ | (2.1.5.2)  \_\_\_\_\_ | (2.1.5.3.1)  \_\_\_\_\_ | (2.1.5.3.2)  \_\_\_\_\_ | (2.1.5.3.3)  \_\_\_\_\_ | (2.1.5.3.4)  \_\_\_\_\_ | |
| PRQ By Language  \_\_\_\_\_ (/10) | (2.1.6.1)  \_\_\_\_\_ | (2.1.6.2)  \_\_\_\_\_ | (2.1.6.3.1)  \_\_\_\_\_ | (2.1.6.3.2)  \_\_\_\_\_ | (2.1.6.3.3)  \_\_\_\_\_ | (2.1.6.3.4)  \_\_\_\_\_ | |
| Student’s Choice  \_\_\_\_\_ (/10) | (2.1.7.1)  \_\_\_\_\_ | (2.1.7.2)  \_\_\_\_\_ | (2.1.7.3.1)  \_\_\_\_\_ | (2.1.7.3.2)  \_\_\_\_\_ | (2.1.7.3.3)  \_\_\_\_\_ | (2.1.7.3.4)  \_\_\_\_\_ | |
|  | 2.2 Details for each of the Post Release Quality History graphs | | | | | | |
| \_\_\_\_\_\_\_ (/2) | 2.2.1 Overview of Post Release Quality History (PRQH) graphs (2) | | | | | | |
|  | 2.2.2, 2.2.3 Details of Individual Post Release Quality graphs (10 points each) | | | | | | |
| Graph  (2.2.x 1)  (2 points) | Analysis & Discussion  (2.2.x.2)  (2 points) | Procedure Used to Produce the Graph (2.2.x.3) | | | | |
| Base Metrics Collected  (2.2.x.3.1)  (1 point) | Compound Metrics  (2.2.x.3.2)  (1 point) | Data Refinement  (2.2.x.3.3)  (2 points) | How to Interpret  (2.2.x.3.4)  (2 points) | |
| PRQH by Quarter  (2.2.2)  \_\_\_\_\_ (/10) | (2.2.2.1)  \_\_\_\_\_ | (2.2.2.2)  \_\_\_\_\_ | (2.2.2.3.1)  \_\_\_\_\_ | (2.2.2.3.2)  \_\_\_\_\_ | (2.2.2.3.3)  \_\_\_\_\_ | (2.2.2.3.4)  \_\_\_\_\_ | |
| PRQH by Year  (2.2.3)  \_\_\_\_\_ (/10) | (2.2.3.1)  \_\_\_\_\_ | (2.2.3.2)  \_\_\_\_\_ | (2.2.3.3.1)  \_\_\_\_\_ | (2.2.3.3.2)  \_\_\_\_\_ | (2.2.3.3.3)  \_\_\_\_\_ | (2.2.3.3.4)  \_\_\_\_\_ | |
| \_\_\_\_\_\_\_  (/100) | Total Assignment Grade | | | | | | |

**Defect Analysis Report**

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

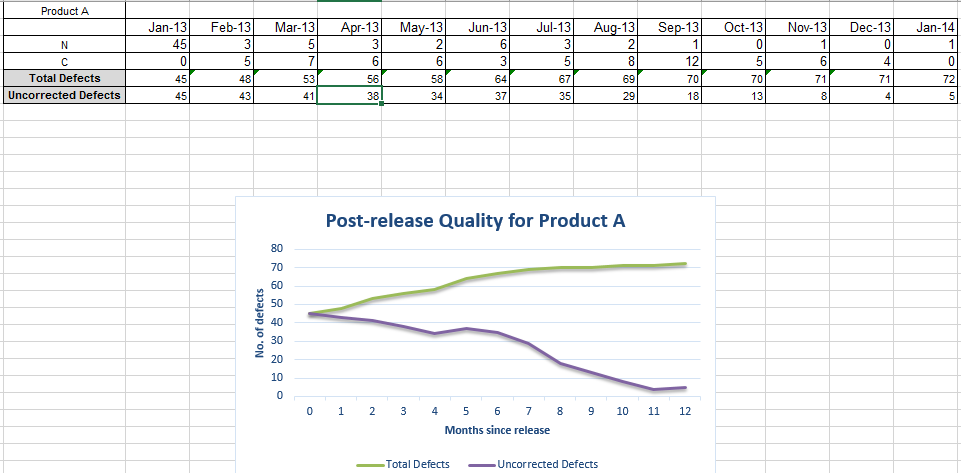
This report shows the results of analyzing three years of defect data on our released products. The purpose is to gain a greater understanding of the quality levels of our released products and to determine whether there is any correlation between software quality and other factors such as the programming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

* 1. **Data Used**

The data necessary to perform these measurements have been collected monthly for each active software product, over the past three years. The data are stored in the **data spreadsheet**, named A2data.xlsx. Figure 1. gives information regarding all the 30 products, their date of release, development process involved, program language used and size of the respective product.

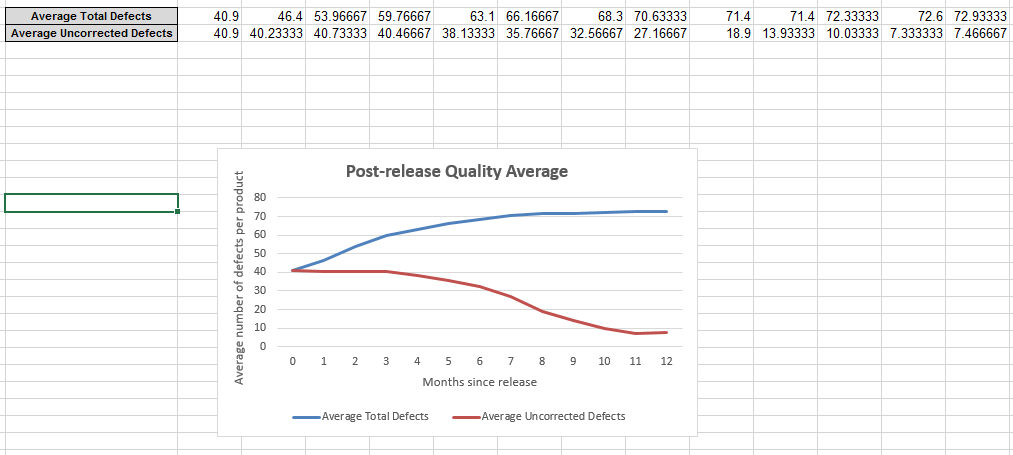
# 

**Figure 1: A2data.xslx**

****

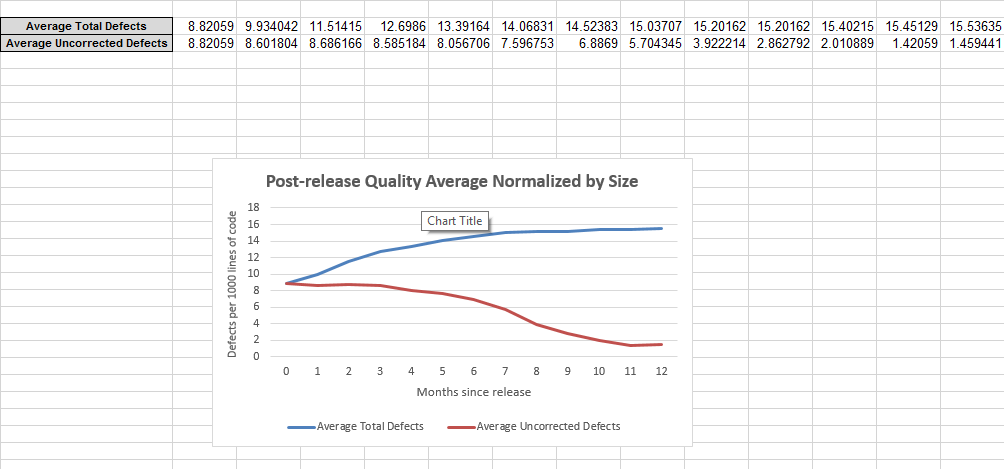
**Figure 2: Post Release Quality for Product A.xslx**

Figure 2. shows graph displaying Post Release Quality for Product A by calculating total and uncorrected defects for the duration of 12 months.

****

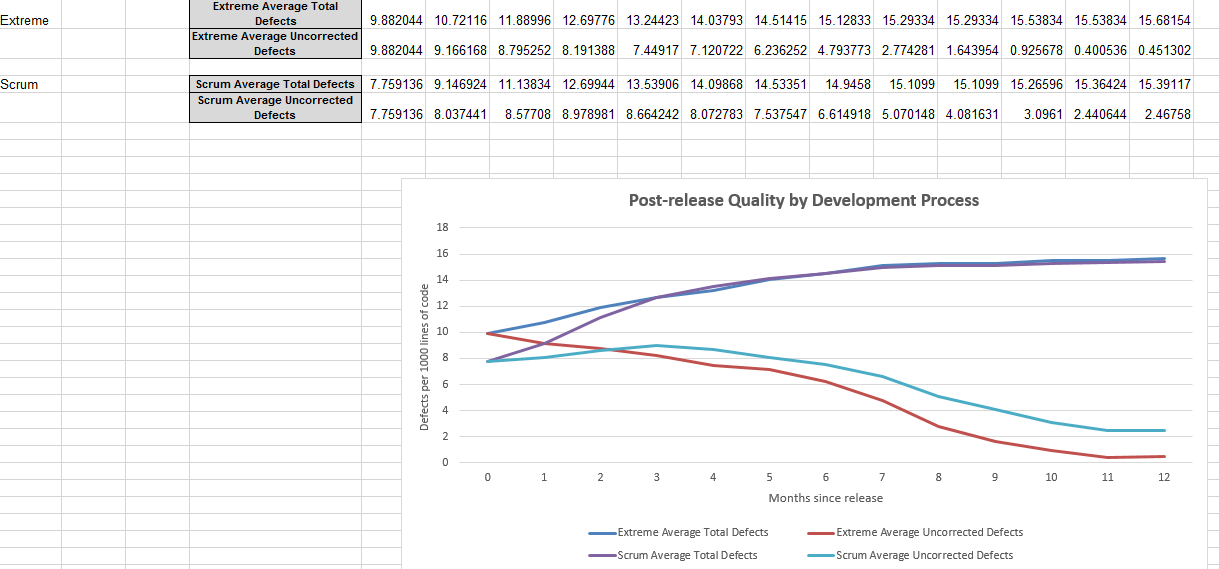
**Figure 3: Post Release Quality Average.xslx**

Figure 3. graph displays the average (mean) for all 30 products by using total defects and uncorrected defects as measures.

****

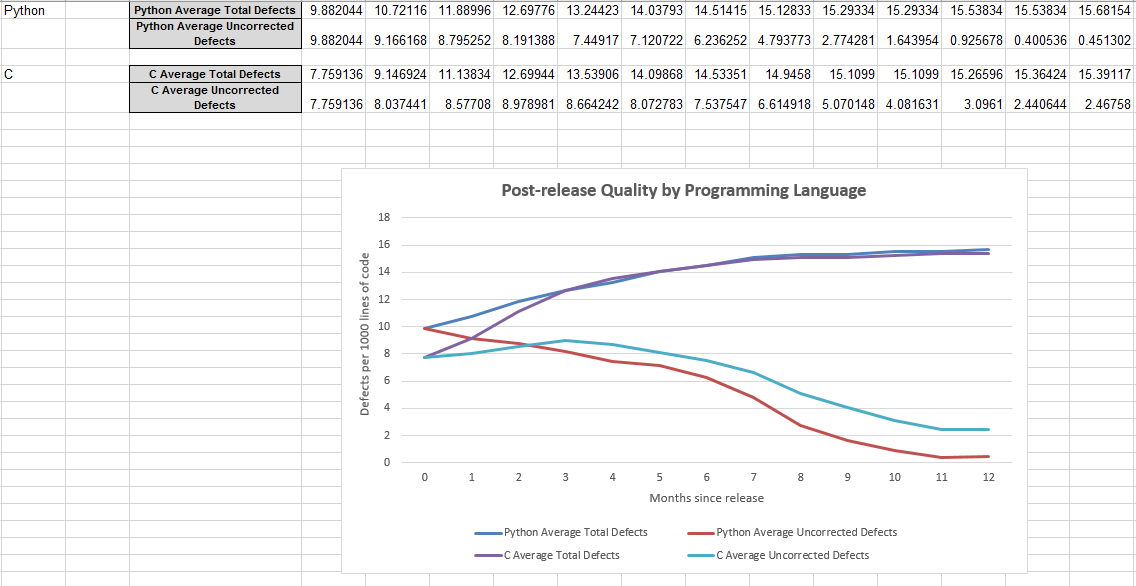
**Figure 4: Post Release Quality Average Normalized by size.xslx**

Figure 4. spreadsheet displays the average (mean) for all 30 products normalized based on the size of the product by using total defects and uncorrected defects as measures.



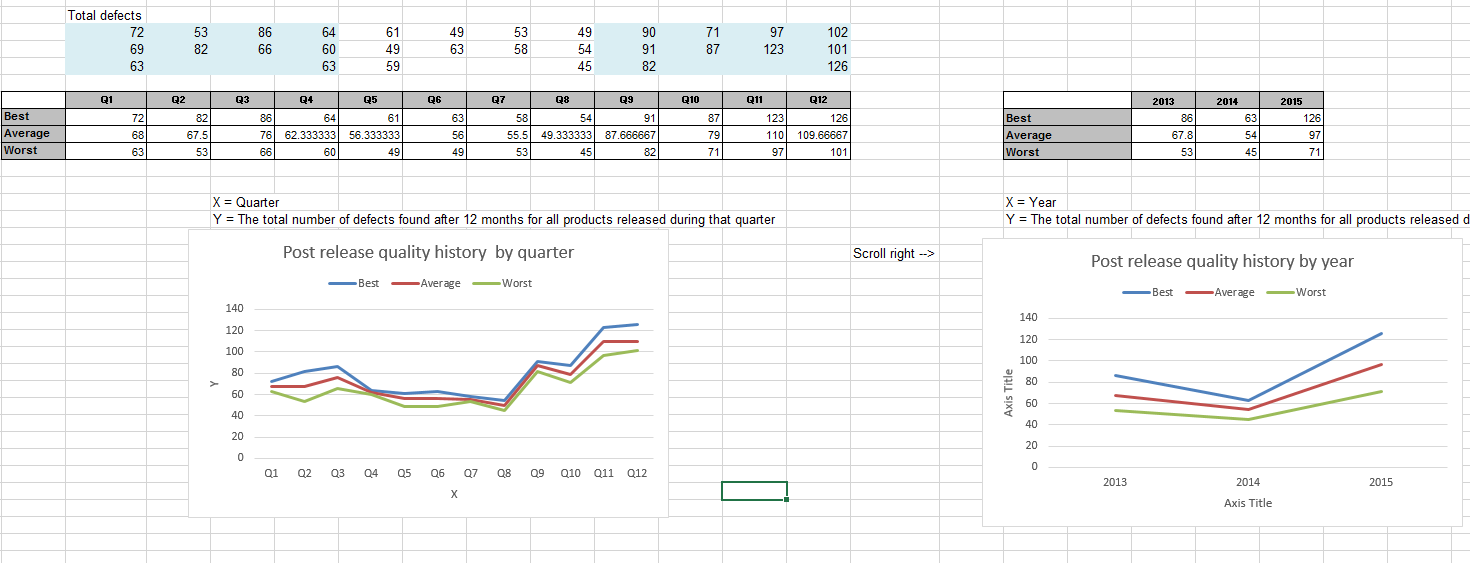
**Figure 5: Post Release Quality Average by Development Process.xslx**

Figure 5. spreadsheet is like previous spreadsheet, but it is categorized based on development process- Extreme and Scrum.



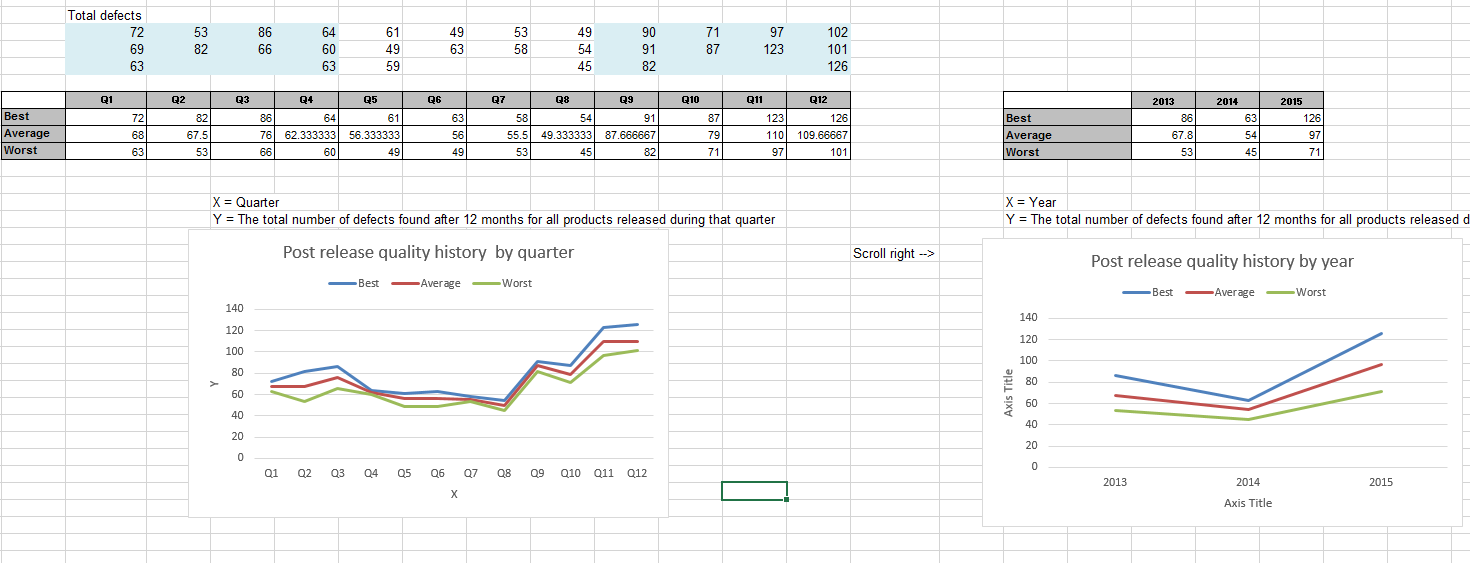
**Figure 6: Post Release Quality Average by Programming Language.xslx**

Figure 6. spreadsheet is like Figure 4. spreadsheet, but it is categorized based on programming language- Python and C.



**Figure 7: Post Release Quality History.xslx - REPLACE**

Figure 8. spreadsheet displays two graphs describing how good were the products we shipped at different times in terms of defect rates.



**Figure 8: Post Release Quality History.xslx**

Figure 8. spreadsheet displays two graphs describing how good were the products we shipped at different times in terms of defect rates.

* 1. **Analyses and Graphs**

We have analyzed the data in several different ways, resulting in two different metrics and their corresponding graphs:

1. Post Release Quality (shown six ways),
2. Post Release Quality History (shown two ways).

These are described in further detail in section 2 of this report.

* 1. **Structure of Analysis Tool**

To analyze defect data, we have created a workbook containing nine spreadsheets. Initial worksheet (Figure 1) gives information of all the 30 products. This data is then refined and analyzed using two measures – total defects and total uncorrected defects. Post Release Quality for Product A.xlxs (Figure 2) worksheet shows one year of history for product A. Post Release Quality Average.xlxs (Figure 3) worksheet show the average (mean) for all 30 products. Post Release Quality Average Normalized by size.xlxs (Figure 4) shows the average for all 30 products, but this time normalized by size. Post Release Quality by Development Process (Figure 5) worksheet calculates total number of defects and total number of uncorrected defects and categorize them based on two groups – Extreme and Scrum. Post Release Quality by Programming Language (Figure 6) worksheet calculates total number of defects and total number of uncorrected defects and categorize them based on two groups – Python and C. Post Release Quality History (Figure 8) worksheet displays two graphs – based on the quarter and based on the month.

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

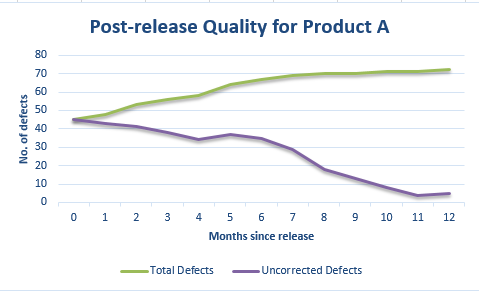
Post release quality is measured for an individual product or for any collection of products (such as all those written in Python) and is defined as the number of defects in the product or collection after release. It answers the question on how many known defects are in this product and how does the total grow after release. This is done by considering the duration of 12 months after the release of product. Two measures – **total defects and total uncorrected defects**, are displayed using line graph.

* + 1. **Post Release Quality for Product A**

Here, the graph show one year of history for product A. The period of 12 months is considered to calculate total defects and total uncorrected defects for product A. Post Release Quality trend is deduced from the data.

**2.1.2.1 Graph**

The graph below shows Post Release Quality for one year of Product A.

****

**Figure 6: Post Release Quality for Product A**

The horizontal axis shows months since release, starting at 0 (the month of release) and continuing to month 12. The vertical axis displays the number of defects ranging from 0 to 80.

**2.1.2.2 Analysis and Discussion**

Figure 6 displays the post-release quality graph for product A having two lines – total number of defects and total uncorrected defects. From the above graph, we can conclude that total number of defects went on increasing gradually. At the time of release total number of defects were 45 which increased to 75 in last month. The total uncorrected defects at the time of release were 45 which decreased drastically to 5 in last month. We observed that, the quality of product A is good, because there was huge decrease in total uncorrected defects post release. But, there was steady growth in total number of defects.

**2.1.2.3 Procedure Used to Collect and Refine Data and Produce Graph**

Systematic approach was followed to collect and refine data as well as to produce Post Release Quality for Product A graph.

|  |
| --- |
| 2.1.2.3.1 Data Collection: Base Metrics Collected |
| The data required for this measure are:   * **DPRE** – The number of known defects at the time of product release. Collected at time of product release. * **DRPT, i** – The number of defects reported in the ***ith*** customer failure report. Collected at the beginning of each month. * **N** – The total number of customer failure reports (total number of months). This is normally 12 for each product. * **DC,i** – The number of defects corrected in month ***i***, reported monthly by engineering staff.   The above data are collected separately for each software product. |

|  |
| --- |
| 2.1.2.3.2 Compound Metrics (Metrics Computed) |
| * **Total Defects** at month T (is defined as the cumulative sum of all defects known at month T. This measure is computed every month, for each product. This includes both defects reported by the customer (post-release defects) and defects known to be in the product at release time (pre-release defects):   Equation - Total Defects   * **Uncorrected Defects** (at month T is defined as **Total Defects** minus the cumulative sum of all defects that have been corrected:   Equation - Uncorrected Defects |

|  |
| --- |
| 2.1.2.3.3 Data Refinement (manipulations, extractions, sorting, etc.) |
| * For this graph, the total defects are computed each month by adding the latest month’s defects to the previous total. * The uncorrected defects are computed each month by subtracting the number of defects corrected in the latest month from the previous uncorrected defects total. |

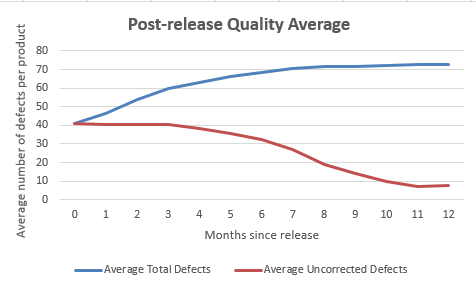
|  |
| --- |
| 2.1.2.3.4 How to Interpret the Graph |
| * Two lines are shown: both DT and DUNC,T are plotted monthly on a line chart for the first release of product A. The horizontal axis is number of months since product release and the vertical axis shows defect count. The chart shows one year of data. * The DT line is monotonic – it will tend to grow slowly, but to flatten out over time, as the product’s defects are found. When the line flattens, it is approximately equal to the total number of defects in the product. * The DUNC,T line should decrease over time, as defects are found and corrected. However it may increase in any particular month if a lot of new defects are found in that month. As we can see from the graph, there was slight increase of total number of uncorrected in 5th month. |

* + 1. **Post Release Quality Average**

Here, the graph shows the average (mean) for all 30 products. There is a comparison of product with the average in that month.

**2.1.3.1 Graph**

The graph below shows Post Release Quality Average for all 30 products.



**Figure 7: Post Release Quality Average**

The horizontal axis shows months since release, starting at 0 (the month of release) and continuing to month 12. The vertical axis displays the average number of defects per product ranging from 0 to 80.

**2.1.3.2 Analysis and Discussion**

Figure 7 displays the post-release quality average graph having two lines – total number of defects and total uncorrected defects. From the above graph, we can conclude, on an average total number of defects went on increasing gradually. At the time of release average total number of defects were 40.99 which increased to 72.93 in last month. The average total uncorrected defects at the time of release were 40.99 which decreased drastically to 7.46 in last month. We observed that, the quality of products is good, because there was tremendous reduction in average total uncorrected defects post release. But, there was steady growth in total number of defects respect to average of all the 30 products.

**2.1.3.3 Procedure Used to Collect and Refine Data and Produce Graph**

Systematic approach was followed to collect and refine data as well as to produce Post Release Quality Average graph.

|  |
| --- |
| 2.1.3.3.1 Data Collection: Base Metrics Collected |
| The data required for this measure are:   * **DPRE** – The number of known defects at the time of product release. Collected at time of product release. * **DRPT, i** – The number of defects reported in the ***ith*** customer failure report. Collected at the beginning of each month. * **N** – The total number of customer failure reports (total number of months). This is normally 12 for each product. * **DC,i** – The number of defects corrected in month ***i***, reported monthly by engineering staff. * **n** – The number of products.   The above data are collected separately for each software product. |

|  |
| --- |
| 2.1.3.3.2 Compound Metrics (Metrics Computed) |
| * **Average Total Defects** at month T (is defined as average number of total defects per product. This measure is computed as the cumulative average of all defects known at month of release for each product. This includes both defects reported by the customer (post-release defects) and defects known to be in the product at release time (pre-release defects):   Equation – Average Total Defects   * **Average Uncorrected Defects** (at month T is defined as **Average** **Total Defects** minus average of corrected defects for all the products:   Equation – Average Uncorrected Defects |

|  |
| --- |
| 2.1.3.3.3 Data Refinement (manipulations, extractions, sorting, etc.) |
| * For this graph, the average total defects are computed each month by adding the latest month’s defects to the previous total for all the products and then dividing by total number of products(n). * The average uncorrected defects are computed each month by subtracting the number of defects corrected in the latest month from the previous uncorrected defects total for all the products and then dividing by total number of products(n). |

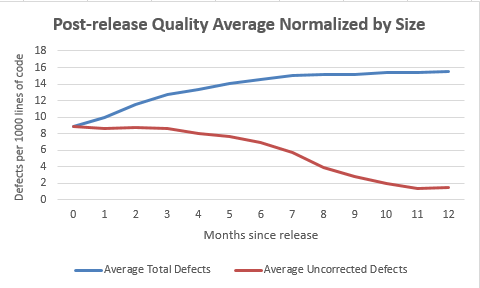
|  |
| --- |
| 2.1.3.3.4 How to Interpret the Graph |
| * Two lines are shown: both DT and DUNC,T are plotted monthly on a line chart for the first release of all the products average. The horizontal axis is number of months since products release and the vertical axis is average number of defects per product. The chart shows one year of data. * The DT line is monotonic – it will tend to grow slowly, but to flatten out over time, as the average of product’s defects are found. When the line flattens, it is approximately equal to the average total number of defects respect to all the products. * The DUNC,T line should decrease over time, as average defects are found and corrected. However, it may increase in any month if a lot of new average defects are found in that month. |

* + 1. **Post Release Quality Average Normalized by Size**

Here, the graph show the average for all 30 products, as above, but this time normalized by size. Defects per 1000 lines of code is calculated using this graph.

**2.1.4.1 Graph**

The graph below shows Post Release Quality Average Normalized by Size for all 30 products.

.

**Figure 8: Post Release Quality Average Normalized by Size**

The horizontal axis shows months since release, starting at 0 (the month of release) and continuing to month 12. The vertical axis displays the average number of defects per 1000 lines of code for all the products.

**2.1.4.2 Analysis and Discussion**

The main reason for normalization is to find average number of defects per 1000 lines of code. Figure 8 displays the post-release quality average normalized by size graph having two lines – average total number of defects and average uncorrected defects. From the above graph, we can conclude, on an average total number of defects went on increasing gradually. At the time of release defects per 1000 lines of code for the average of all 30 products was 8.82. As we can see, after 12 months, this number increased 15.53. Thus, average total number of defects increased gradually within a period of 12 months. The average uncorrected defects at the time of release were 8.82 which were decreased to 1.45 in last month. We observed that, defects per 1000 lines of code for average of all products went on increasing. But, in the same time the average uncorrected defects declined tremendously.

**2.1.4.3 Procedure Used to Collect and Refine Data and Produce Graph**

Systematic approach was followed to collect and refine data as well as to produce Post Release Quality Average Normalized by Size graph.

|  |
| --- |
| 2.1.4.3.1 Data Collection: Base Metrics Collected |
| The data required for this measure are:   * **DPRE** – The number of known defects at the time of product release. Collected at time of product release. * **DRPT, i** – The number of defects reported in the ***ith*** customer failure report. Collected at the beginning of each month. * **N** – The total number of customer failure reports (total number of months). This is normally 12 for each product. * **DC,i** – The number of defects corrected in month ***i***, reported monthly by engineering staff. * **DS** – The total defects per 1000 lines of code for a product. * **DC,i,s** – The number of defects corrected in month ***i divided by size of the product multiplied by 1000***, reported monthly by engineering staff. * **n** – The number of products. * **s** – Size of the product at the time of release.   The above data are collected separately for each software product. |

|  |
| --- |
| 2.1.4.3.2 Compound Metrics (Metrics Computed) |
| * **Average Total Defects** at month T (is defined as average number of total defects per product normalized by size. This measure is computed as dividing each product’s defect counts by the product’s size (and then multiply by 1000) to produce defects per 1000 lines of code. Then compute the cumulative average of all defects known at month of release for all products. This includes both defects reported by the customer (post-release defects) and defects known to be in the product at release time (pre-release defects):   Equation – Total Defects per 1000 lines of code for product  Equation 2 – Average Total Defects   * **Average Uncorrected Defects** (at month T is defined as **Average** **Total Defects** minus average of corrected defects for all the products normalized by size.   Equation 3 – Uncorrected Defects per 1000 lines of code for product  Equation 4 – Average Uncorrected Defects |

|  |
| --- |
| 2.1.4.3.3 Data Refinement (manipulations, extractions, sorting, etc.) |
| * For this graph, the average total defects are computed each month by dividing each product’s defect counts by the product’s size and then multiply by 1000 for all the 30 products and then taking the average. * The average uncorrected defects are computed each month by dividing each product’s corrected defect counts by the product’s size and then multiply by 1000 for all the 30 products and then subtracting the number of defects corrected in the latest month from the previous uncorrected defects total for all the products. |

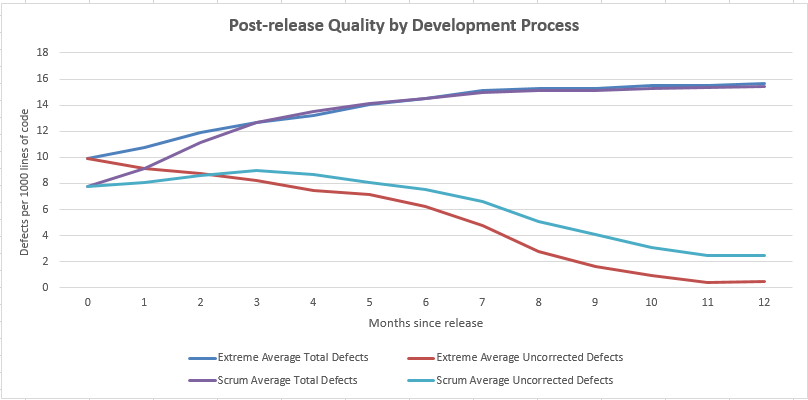
|  |
| --- |
| 2.1.4.3.4 How to Interpret the Graph |
| * Two lines are shown: both DT and DUNC,T are plotted monthly on a line chart for the first release of all the products average normalized by size. The horizontal axis is number of months since products release and the vertical axis is average number of defects divided by size per product. The chart shows one year of data. * The DT line is monotonic – it will tend to grow slowly, but to flatten out over time, as the average of product’s defects are found. When the line flattens, it is approximately equal to defects per 1000 lines of code respect to all the products. * The DUNC,T line should decrease over time, as average defects are corrected and divided by size of product. The line flattens after 11 months, since the uncorrected defects remains constant. |

* + 1. **Post Release Quality by Development Process**

Here, the graph show the average for all 30 products normalized by size and categorized by development process – Scrum and Extreme Programming. Defects per 1000 lines of code is calculated using this graph.

**2.1.5.1 Graph**

The graph below shows Post Release Quality by Development Process for each of the two processes used: Extreme Programming and SCRUM.



**Figure 8: Post Release Quality by Development Process**

**2.1.5.2 Analysis and Discussion**

From the above graph, we can clearly make out that average uncorrected defects for Extreme process reduced gradually over a period of 12 months. Whereas, there was significant rise in average uncorrected defects for Scrum process from 2nd month to 3rd month. In last month, average total defects for both the process were almost similar – 15.68 for Extreme and 15.32 for Scrum.

**2.1.5.3 Procedure Used to Collect and Refine Data and Produce Graph**

Systematic approach was followed to collect and refine data as well as to produce Post Release Quality Average by Development graph.

|  |
| --- |
| 2.1.5.3.1 Data Collection: Base Metrics Collected |
| The data required for this measure are:   * **DPRE** – The number of known defects at the time of product release. Collected at time of product release. * **DRPT, i** – The number of defects reported in the ***ith*** customer failure report. Collected at the beginning of each month. * **N** – The total number of customer failure reports (total number of months). This is normally 12 for each product. * **DC,i** – The number of defects corrected in month ***i***, reported monthly by engineering staff. * **DS** – The total defects per 1000 lines of code for a product. * **DC,i,s** – The number of defects corrected in month ***i divided by size of the product multiplied by 1000***, reported monthly by engineering staff. * **n** – The number of products. * **s** – Size of the product at the time of release.   The above data are collected separately for each software product. |

|  |
| --- |
| 2.1.5.3.2 Compound Metrics (Metrics Computed) |
| **Extreme Programming Development Process**   * **Average Total Defects** at month T (is defined as average number of total defects per product normalized by size. This measure is computed as dividing each product’s defect counts by the product’s size (and then multiply by 1000) to produce defects per 1000 lines of code. Then compute the cumulative average of all defects known at month of release for all products developed using Extreme Programming. This includes both defects reported by the customer (post-release defects) and defects known to be in the product at release time (pre-release defects):   Equation – Total Defects per 1000 lines of code for product  Equation 2 – Average Total Defects   * **Average Uncorrected Defects** (at month T is defined as **Average** **Total Defects** minus average of corrected defects for all the products developed using Extreme Programming normalized by size.   Equation 3 – Uncorrected Defects per 1000 lines of code for product  Equation 4 – Average Uncorrected Defects  **Scrum Development Process**   * **Average Total Defects** at month T (is defined as average number of total defects per product normalized by size. This measure is computed as dividing each product’s defect counts by the product’s size (and then multiply by 1000) to produce defects per 1000 lines of code. Then compute the cumulative average of all defects known at month of release for all products developed using Scrum Development Process. This includes both defects reported by the customer (post-release defects) and defects known to be in the product at release time (pre-release defects):   Equation – Total Defects per 1000 lines of code for product  Equation 2 – Average Total Defects   * **Average Uncorrected Defects** (at month T is defined as **Average** **Total Defects** minus average of corrected defects for all the products developed using Scrum Development process normalized by size.   Equation 3 – Uncorrected Defects per 1000 lines of code for product  Equation 4 – Average Uncorrected Defects |

|  |
| --- |
| 2.1.5.3.3 Data Refinement (manipulations, extractions, sorting, etc.) |
| * For this graph, the average total defects are computed each month by dividing each product’s defect counts by the product’s size and then multiply by 1000 for all the products developed by that development process (Extreme and Scrum) and then taking the average of the group. * The average uncorrected defects are computed each month by dividing each product’s corrected defect counts by the product’s size and then multiply by 1000 for all the products developed by that development process (Extreme and Scrum) and then subtracting the number of defects corrected in the latest month from the previous uncorrected defects total for all the products of that group. |

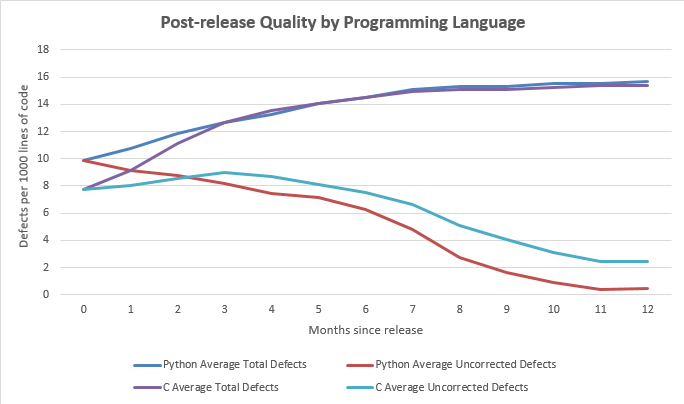
|  |
| --- |
| 2.1.5.3.4 How to Interpret the Graph |
| * Four lines are shown: DT,S , DT,E and DUNC,TE , DUNC,TS  are plotted monthly on a line chart for the first release of all the products average developed by that development process (Extreme and Scrum) normalized by size. The horizontal axis is number of months since products release and the vertical axis is average number of defects divided by size per product grouped into development process. The chart shows one year of data. * Both DTE and DTS line is monotonic – it will tend to grow slowly, but to flatten out over time, as the average of product’s defects are found. When the line flattens, it is approximately equal to defects per 1000 lines of code respect to all the products of that group. * Both DUNC,TE and DUNC,TS line should decrease over time, as average defects are corrected and divided by size of product of that group. The line flattens after 11 months, since the uncorrected defects remains constant for both groups. |

* + 1. **Post Release Quality by Programming Language**

Here, the graph show the average for all 30 products normalized by size and categorized by development process – Scrum and Extreme Programming. Defects per 1000 lines of code is calculated using this graph.

**2.1.6.1 Graph**

The graph below shows Post Release Quality by programming language for each of the two languages used: Python and C.



**Figure 9: Post Release Quality by Programming Language**

**2.1.6.2 Analysis and Discussion**

From the above graph, we can clearly make out that average uncorrected defects for Python programming approach reduced gradually over a period of 12 months. Whereas, there was significant rise in average uncorrected defects for C programming approach from 2nd month to 3rd month. In last month, average total defects for both the process were almost similar – 15.69 for Python and 15.39 for C.

**2.1.6.3 Procedure Used to Collect and Refine Data and Produce Graph**

Systematic approach was followed to collect and refine data as well as to produce Post Release Quality by Programming Language graph.

|  |
| --- |
| 2.1.6.3.1 Data Collection: Base Metrics Collected |
| The data required for this measure are:   * **DPRE** – The number of known defects at the time of product release. Collected at time of product release. * **DRPT, i** – The number of defects reported in the ***ith*** customer failure report. Collected at the beginning of each month. * **N** – The total number of customer failure reports (total number of months). This is normally 12 for each product. * **DC,i** – The number of defects corrected in month ***i***, reported monthly by engineering staff. * **DS** – The total defects per 1000 lines of code for a product. * **DC,i,s** – The number of defects corrected in month ***i divided by size of the product multiplied by 1000***, reported monthly by engineering staff. * **n** – The number of products. * **s** – Size of the product at the time of release.   The above data are collected separately for each software product. |

|  |
| --- |
| 2.1.6.3.2 Compound Metrics (Metrics Computed) |
| **Python Programming Language**   * **Average Total Defects** at month T (is defined as average number of total defects per product normalized by size. This measure is computed as dividing each product’s defect counts by the product’s size (and then multiply by 1000) to produce defects per 1000 lines of code. Then compute the cumulative average of all defects known at month of release for all products developed using Python Programming Language. This includes both defects reported by the customer (post-release defects) and defects known to be in the product at release time (pre-release defects):   Equation 1 – Total Defects per 1000 lines of code for product  Equation 2 – Average Total Defects   * **Average Uncorrected Defects** (at month T is defined as **Average** **Total Defects** minus average of corrected defects for all the products developed using Extreme Programming normalized by size.   Equation 3 – Uncorrected Defects per 1000 lines of code for product  Equation 4 – Average Uncorrected Defects  **C Programming Language**   * **Average Total Defects** at month T (is defined as average number of total defects per product normalized by size. This measure is computed as dividing each product’s defect counts by the product’s size (and then multiply by 1000) to produce defects per 1000 lines of code. Then compute the cumulative average of all defects known at month of release for all products developed using C Programming Language. This includes both defects reported by the customer (post-release defects) and defects known to be in the product at release time (pre-release defects):   Equation 1 – Total Defects per 1000 lines of code for product  Equation 2 – Average Total Defects   * **Average Uncorrected Defects** (at month T is defined as **Average** **Total Defects** minus average of corrected defects for all the products developed using Scrum Development process normalized by size.   Equation 3 – Uncorrected Defects per 1000 lines of code for product  Equation 4 – Average Uncorrected Defects |

|  |
| --- |
| 2.1.6.3.3 Data Refinement (manipulations, extractions, sorting, etc.) |
| * For this graph, the average total defects are computed each month by dividing each product’s defect counts by the product’s size and then multiply by 1000 for all the products developed by that programming language (Python and C) and then taking the average of the group. * The average uncorrected defects are computed each month by dividing each product’s corrected defect counts by the product’s size and then multiply by 1000 for all the products developed by that programming language (Python and C) and then subtracting the number of defects corrected in the latest month from the previous uncorrected defects total for all the products of that group. |

|  |
| --- |
| 2.1.6.3.4 How to Interpret the Graph |
| * Four lines are shown: DT,C , DT,P and DUNC,TC , DUNC,TP  are plotted monthly on a line chart for the first release of all the products average developed by that programing language (Python or C) normalized by size. The horizontal axis is number of months since products release and the vertical axis is average number of defects divided by size per product grouped per programming language. The chart shows one year of data. * Both DTP and DTC line is monotonic – it will tend to grow slowly, but to flatten out over time, as the average of product’s defects are found. When the line flattens, it is approximately equal to defects per 1000 lines of code respect to all the products of that group. * Both DUNC,TP and DUNC,TC line should decrease over time, as average defects are corrected and divided by size of product of that group. The line flattens after 11 months, since the uncorrected defects remains constant for both groups. |

* + 1. **????? (Your choice here)**

Provide a brief description of this measure. This is similar to the overview in 2.1.1, except that in this case it is specific to this graph rather than to the collection of post release quality graphs.

**2.1.7.1 Graph**

The graph below shows ?????

Insert your own figure here, along with additional descriptions if you wish.

**2.1.7.2 Analysis and Discussion**

This section is for analysis, not just description. Here you analyze and describe the graph shown above and what conclusions or observations you make from looking at the graph. In this case, be sure to discuss what your graph shows.

**2.1.7.3 Procedure Used to Collect and Refine Data and Produce Graph**

In the tables below, explain the details of your data refinement and how you produce the graph. The purpose of this section is to enable someone else to start with the same data and produce the same graphs. Describe each step of the measurement process: data collected, data refinement performed, compound measures calculated, and what data are shown on the graph. A sample of what is expected is shown below. You can use this for the first graph but need to provide the corresponding information for all other graphs.

|  |
| --- |
| 2.1.7.3.1 Data Collection: Base Metrics Collected |
| Here you list and describe the base metrics needed to compute this metric. Also describe when the data are collected (how often, etc.) |

|  |
| --- |
| 2.1.7.3.2 Compound Metrics (Metrics Computed) |
| Here you show all compound metrics used, including their formulas and any intermediate computations needed. |

|  |
| --- |
| 2.1.7.3.3 Data Refinement (manipulations, extractions, sorting, etc.) |
| Here you explain all data manipulations and refinements needed to compute this metric, such as computing totals, sorting, shifting data to new columns, and so forth. |

|  |
| --- |
| 2.1.7.3.4 How to Interpret the Graph |
| Here you explain how to interpret the lines, bars or other elements of the graph. |

* 1. **Post Release Quality History**
     1. **Overview**

Post release quality history is measured for a collection of products. It shows whether there is any relationship between the level of quality and when the product was released. This is a “lagging” indicator, which means it can only be computed after a release has been out for 12 months. The measure indicates the total number of defects found for all products in the collection during their first 12 months of use. The best case, worst case and average case were calculated for the products released during the given time.

* + 1. **Post Release Quality History by Quarter**

Here, the graph show the total number of defects found after 12 months for all products released during that quarter. It depicts 3 cases-

1. Best Case: The product released during that quarter having least number of defects.
2. Worst Case: The product released during that quarter having most number of defects.
3. Average Case: It is the average of first two cases respect to products released during that quarter.

**2.2.2.1 Graph**

The graph below shows Post Release Quality History by Quarter

**Figure 11: Post Release Quality History by Quarter**

**2.2.2.2 Analysis and Discussion**

From the graph shown above, we can easily make out that there was significant drop in total number of defects of all the products released during 8th quarter. In first quarter, the best case is 63 total number of defects of all the products, worst case is 72 and average case is 68 total number of defects of all the products. After 8th quarter, total number of defects drastically increased for subsequent quarters. The most number of total defects can be seen in last quarter.

**2.2.2.3 Procedure Used to Collect and Refine Data and Produce Graph**

Systematic approach was followed to collect and refine data as well as to produce Post Release Quality by Programming Language graph.

|  |
| --- |
| 2.2.2.3.1 Data Collection: Base Metrics Collected |
| The data required for this measure are:   * **DPRE** – The number of known defects at the time of product release. Collected at time of product release. * **DRPT, i** – The number of defects reported in the ***ith*** customer failure report. Collected at the beginning of each month. * **N** – The total number of customer failure reports (total number of months). This is normally 12 for each product. * **DC,i** – The number of defects corrected in month ***i***, reported monthly by engineering staff.   The above data are collected separately for each software product. |

|  |
| --- |
| 2.2.2.3.2 Compound Metrics (Metrics Computed) |
| **The total number of defects found after 12 months for all products released during that quarter**   * **Total Defects** at month T (is defined as the cumulative sum of all defects known at month T. This measure is computed every month, for each product. This includes both defects reported by the customer (post-release defects) and defects known to be in the product at release time (pre-release defects):   Equation 1 - Total Defects |

|  |
| --- |
| 2.2.2.3.3 Data Refinement (manipulations, extractions, sorting, etc.) |
| * For this graph, the total defects are computed each month by adding the latest month’s defects to the previous total. * The above calculation is done for all the products released during that quarter. * Then, best, worst and average case is found by sorting total defects calculated per product. |

|  |
| --- |
| 2.2.2.3.4 How to Interpret the Graph |
| * Three lines are shown: all depicting DT are plotted quarterly on a line chart for the release of all the products in that quarter categorized into best, worst and average case. The horizontal axis is quarter in which product was release and the vertical axis shows total defect count. The chart shows three year of data. * The DT line is monotonic – it will tend to grow slowly, but to flatten out over time, as the product’s defects are found in that quarter. When the line flattens, it is approximately equal to the total number of defects in the product. But, in the above graph, is fluctuating from high in one quarter and low in other, vice versa. |

* + 1. **Post Release Quality History by Year**

Here, the graph show the total number of defects found after 12 months for all products released during that year. It depicts 3 cases-

1. Best Case: The product released during that year having least number of defects.
2. Worst Case: The product released during that year having most number of defects.
3. Average Case: It is the average of first two cases respect to products released during that year.

**2.2.3.1 Graph**

The graph below shows Post Release Quality History by Year

**Figure 11: Post Release Quality History by Year**

**2.2.3.2 Analysis and Discussion**

This section is for analysis, not just description. Here you analyze and describe the graph shown above and what conclusions or observations you make from looking at the graph. In this case, be sure to discuss how graphing by year differs from graphing by quarter and what the relative benefits are for each approach.

**2.2.3.3 Procedure Used to Collect and Refine Data and Produce Graph**

In the tables below, explain the details of your data refinement and how you produce the graph. The purpose of this section is to enable someone else to start with the same data and produce the same graphs. Describe each step of the measurement process: data collected, data refinement performed, compound measures calculated, and what data are shown on the graph. A sample of what is expected is shown below. You can use this for the first graph but need to provide the corresponding information for all other graphs.

|  |
| --- |
| 2.2.3.3.1 Data Collection: Base Metrics Collected |
| Here you list and describe the base metrics needed to compute this metric. Also describe when the data are collected (how often, etc.) |

|  |
| --- |
| 2.2.3.3.2 Compound Metrics (Metrics Computed) |
| Here you show all compound metrics used, including their formulas and any intermediate computations needed. |

|  |
| --- |
| 2.2.3.3.3 Data Refinement (manipulations, extractions, sorting, etc.) |
| Here you explain all data manipulations and refinements needed to compute this metric, such as computing totals, sorting, shifting data to new columns, and so forth. |

|  |
| --- |
| 2.2.3.3.4 How to Interpret the Graph |
| Here you explain how to interpret the lines, bars or other elements of the graph. |