SOEN6611 – summer term 2022

Assignment 1 on simple measurement data collection and analysis (SLOC, Effort, #Defects, Productivity).

Weight: 5%, individual work

Posted on 12/05/2022, due on 21/05/2022

Goals:
G1 (2%). Practice data collection on length in physical source lines of code SLOC, Work Effort in person-minutes, #Defects, Estimated Effort.
G2 (1.5%). Derive a productivity indicator based on historical data collected from similar assignments by students at Concordia (RE: ISO 15939 appendix A). Analyze your productivity. What conclusions can your draw?
G3 (1.5%). Derive a defect density indicator based on historical data collected from similar assignments by students at Concordia (RE: ISO 15939 appendix A). Analyze your defect density. What conclusions can your draw?

Detailed descriptions of the goals are provided next:

Goal G1 (2%). Practice measurement data collection and analysis: length in physical source lines of code SLOC, Estimated Effort in person-minutes, Work effort (in person-minutes), and Number of Defects.

Tasks:

1. Read the document LENGTH OF SOURCE CODE BY Dr. PANKAJ KAMTHAN (included in the zipped file. Please note that some of the links are outdated)
2. Estimate the Effort (in person-minutes) that will be required from you to write OO code to calculate the ***distance between two gps coordinates using latitude and longitude in radians*** in a programming language of your choice. Record your Estimated Effort.
   NOTE 1: If your latitude and longitude are in degrees, then divide by 180/PI to convert to radians [radians = degrees * PI / 180].
3. Record the Work Effort [in person-minutes] it actually took you to write and test the program, including finding and correcting bugs.
4. Record the Number of Defects found during the testing of your code.
5. Calculate the length of your program in physical SLOC (source lines of code):
   a. Manually, documenting the rules for the counting of the lines of code that you used, and then
   b. Automatically, using a code length measurement tool of your choice.

6. Interpret the results of 5-a. and 5-b. (are the results identical? If not, why? Is this tool reliable for collecting SLOC data?) Comment on the importance of clearly defined rules in code length measurement.

G1 Deliverables:

- Your source code, Estimated Effort, Work Effort (exact time it took you to write it), and Number of Defects found.
- Your rules for the counting the physical source lines of code SLOC, a reference, to the tool which you used for automatic calculation of code length, and the tool's measurement results (RE: task 5-a. and 5-b)
- Analysis and interpretation of the results (RE: task 6)

G2 (1.5%). Simple analysis of coding productivity based on historical data collected from similar assignments by Concordia students.

G2 Tasks:

1. Calculate the Productivity of each Programmer in the enclosed excel file, which contains historical data on SLOC and work effort data collection from past similar assignments.

   Use the following Measurement Formula for Productivity:
   Productivity of Programmer X = Length of Code of Programmer X / Programmer X Effort
   Plot the productivity measurement results of the historical data on a graph, where **x-axis** represents the programmers and the **y-axis** is the productivity of the corresponding programmers.

2. Compute the mean and standard deviation of the past coding productivity values. Apply Control limits chart to the past coding productivity values. Depict the mean and the control limits on the graph from Task 1.
   NOTE: *The mean is used to model average past productivity of the programmers. Computed upper and lower control limits based on the standard deviation indicate that anything within the control limits should be viewed as expected variation. Anything outside of control limits warrants investigation.*
3. Plot your actual productivity result on the graph from G2 Task 2, interpret it. What conclusions can your draw? Explain your findings.

G2 Deliverables:

- The enclosed excel file updated with the programmers' coding productivity values. G2 Task 1 graph.
- Mean, standard deviation, UCL and LCL of the programmers' coding productivity values (show all calculations). Depict the mean and the control limits on the graph.
- Analyze and interpret the results (RE: task 3)

G3 (1.5%). Simple analysis of defect density based on historical data collected from similar assignments by Concordia students.

G3 Tasks:

1. Calculate the **defect density** of each Programmer in the enclosed excel file, which contains historical data on SLOC and #Defects data collection from past similar assignments.

   Use the following Measurement Formula for **defect density**
   Defect Density of Programmer X = Programmer X # Defects / Programmer X Length of Code
   Plot the **defect density** measurement results of the historical data on a graph, where **x-axis** represents the programmers and the **y-axis** is the **defect density** of the corresponding programmers.

2. Compute the mean and standard deviation of the past **defect density** values. Apply Control limits chart to the past **defect density** values. Depict the mean and the control limits on the graph from Task 1.
   NOTE: *The mean is used to model average past* **defect density** *of the programmers. Computed upper and lower control limits based on the standard deviation indicate that anything within the control limits should be viewed as expected variation. Anything outside of control limits warrants investigation.*
3. Plot your actual **defect density** result on the graph from G3 Task 2 interpret it. What conclusions can your draw? Explain your findings.

G3 Deliverables:

- The enclosed excel file updated with the programmers' **defect density** values. G2 Task 1 graph.
- Mean, standard deviation, UCL and LCL of the programmers' **defect density** values (show all calculations). Depict the mean and the control limits on the graph.
- Analyze and interpret the results (RE: task 3)

Help material for Step 2: What are Control Limits?
Let us understand what you are looking at.
Mean: A statistically calculated number that defines the average amount of variation in your productivity data.
UCL (Upper Control Limit): A statistically calculated number that defines the higher limit of variation in your productivity data.
LCL (Lower Control Limit): A statistically calculated number that defines the lower limit of variation in your productivity data.

How do you compute the Control Limits (UCL & LCL)?

The general rule of thumb for calculating control limits is:

(Average KPI Value) +/- (2 x (Standard Deviation))

Here, control limits are calculated 2 standard deviations above or below the mean of productivity data values.

Population Standard Deviation: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$

NOTE: In a case of a negative lower control limit, LCL will not be useful in detecting special cause variation causing unusually low values and alternative rules must be used

Note for future reference: How to use the control limits in software measurement?

In a software project world where there are tons of measurement data, control limits are extremely helpful in leveraging the power of statistics to be the first filter of when you should dig deeper or look for a cause of deviation. If your measurement data and trends have variations from milestone to millstone this is a great way to isolate what is "normal" and what is "abnormal" in the data trend. Not only that, but if a series of data points fall outside the control limits then it is a bigger red flag in terms of something highly impactful going wrong.