CITS5502 Software Processes

Semester 2, 2019

Assignment 1 – Simulating a process

Nicole Low- 21151969

1. Introduction

Predictability is an integral part of the software development cycle. It is useful for creating estimates, deadlines and development plans in a software project. However, software development is naturally unpredictable due to possible technical and other issues found during project execution.

One way in which software processes can improve predictability is through the use of metrics in the test and repair process. Metrics can provide a way for progress of a project to be measured, allowing comparison to similar projects, which gives a greater sense on the progress of the project, system reliability and arguably thus customer satisfaction. These metrics are tested through simulation; using real data to trial different approaches for a projects test and repair stage.

This report will model a set of 'real world' data, define a set of metrics and assumptions, and develop a simulation model.

2. Assumptions

There are a range of assumptions required for making estimations and simulations within this project, these are;

- t is equal to the number of the current week, $t = 1$ is the first week
- pt is equal to the number of people who are testers in the current week
- pr is equal to the number of people who are repairers in the current week
- $pr + pt = p, p = 3, pr, pt \geq 0$
- pr and pt are constant over a given week
- The data lists defects found each week independent of each other week
- Defects can be corrected in the week they are found
- When there are no criteria differentiating two defects, the oldest defect will be selected first
- "Major or minor", "easy or hard" and "hours to fix" are accurate
- The statement that major defects are "seven times as damaging" on average as minor ones is accurate
- Hard defects will take 5 hours to repair
- Easy defects will take 2 hours to repair
- Employees can be moved between test and repair roles with no efficiency disadvantage

- Employees work at constant and identical efficiency
- n people work n times as fast as 1 person; efficiency increases in a linear trend
- The data assumes the number of defects found per week were found with $pt = 1$
- Increasing pt to n will result in the defects found in weeks wk, ..., $wk+n-1$ being found in week wk
- For any week where $pt = 0$, no defects are found

3. Fitting Defect Detection Curves

3.1 Exponential Function

The number of found defects can be fit to the exponential function $y = ae^{bx}$ and can be used to provide an estimate of the number of defects remaining in the project after the test and repair stage concludes ($t > 20$).

As seen in Figure 1, the values $a = 60.842$, $b = -0.179$, $R^2 = 0.932$ were found by fitting the found defects over the period $t = 20$ using the method of least squares in Microsoft Excel, with a coefficient of determination of approximately 93.2%. This is an okay fit and indicates that the model explains 93.2% of the variability of the response data around its mean.

To calculate the number of defects remaining, the function takes the end of the test and repair stage to infinity;

$$\int_{20}^{\infty} 60.842e^{-0.179x}\,dx = 9.475(3dp) \approx 9$$

Therefore, given the assumption the defects will decrease exponentially, there are approximately 9 defects remaining.

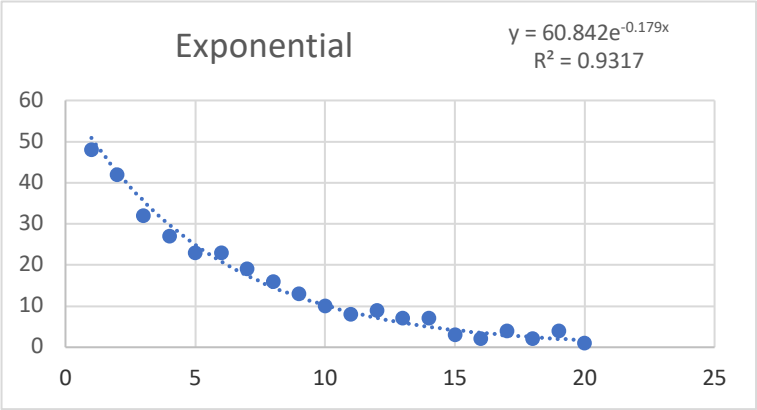

Figure 1: Exponential defect detection curve fit to data

3.2 Linear Regression

The number of found defects can be fit to the linear function $y = a + bx$ and can be used to provide an estimate of the number of defects remaining in the project after the test and repair stage concludes ($t > 20$).

As seen in Figure 2, the values $a = 37.437$, $b = -2.136$, $R^2 = 0.8534$ were found by fitting the found defects over the period $t = 20$ using the method of least squares in Microsoft Excel, with a coefficient of determination of approximately 85.3%. This is an okay fit and indicates that the model explains 85.3% of the variability of the response data around its mean. However, this is a less accurate fit than the exponential function above. This is because evidently the data does not decline in a true linear pattern.

The function estimates that when $t = 18$, $y = -1.0263158$. This means the function predicted that all defects would be located and fixed by week 18 in the test and repair stage.

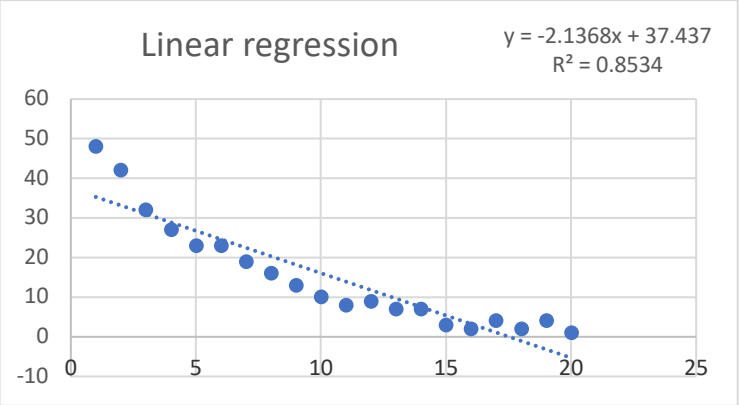

Figure 2: Linear defect detection trendline fit to data

3.3 Polynomial Function

The number of found defects can be fit to the polynomial function $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$ and can be used to provide an estimate of the number of defects remaining in the project after the test and repair stage concludes ($t > 20$).

The function $y = 0.159x^2 - 5.4766x + 49.682$, with an $R^2 = 0.9782$ was found by fitting the found defects over the period $t = 20$ using the method of least squares in Microsoft Excel, with a coefficient of determination of approximately 97.8%. Judging the model off of the R^2 value suggests that this is a good fit and indicates that the model explains 97.8% of the variability of the response data around its mean. However logically the polynomial function does not make sense in the software scenario, as it is unlikely the amount of found defects will rise after week 21.

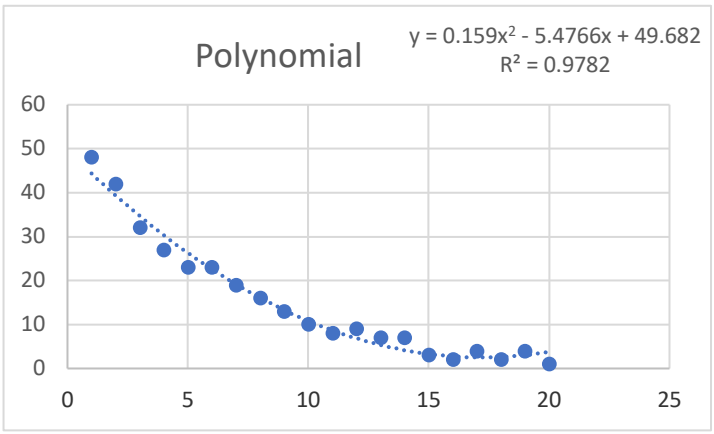

Figure 3: Polynomial defect detection curve fit to data

The most plausible model for the data is the negative exponential curve. Although the polynomial function has a stronger R^2, in a real-life software development scenario, it is unlikely there would be a consistent rise in defects found after 20 weeks of the test and repair stage. The exponential curve is a closer fit to the natural data trend and makes logical sense in a real-life setting.

4. Chosen Metrics and Staff Allocation

Certain staff allocation strategies will cause the chosen metrics to produce different results. The variations of these metrics will be evaluated by how those metrics affect system reliability and customer satisfaction. System reliability will be defined as when a strategy gives little to no defects remaining at the end of the test and repair stage (i.e., when $t = 20$). Customer satisfaction will be defined as when the chosen metric remains at a low level or trends negatively.

Three chosen metrics to investigate are;

a. Total impact of found defects still to be fixed
b. The average time in the “to be fixed” queue of major defects which are still to be fixed
c. The ratio of defects fixed to defects found

Three strategies of prioritising defect detection and repair were run against a model calculating the result of each metric, they are;

a. fix defects in random order
b. fix defects in the order they are found
c. fix major defects first, fix easy defects first

The simulations were made first by using effort allocation programs with Python (effort_allocation.py), which simulate defect detection and repair in a random order, the order they are found, and in a major to minor, easy to hard method. These results were entered into Microsoft Excel, which modelled the results against the chosen metrics.

4.1 Total impact of found defects still to be fixed

Figure 4 illustrates the strategy results for the total importance of found defects still to be fixed. This was calculated as;

$$\sum_{p \in P} pd_p(t)$$

Where P is the set of priorities and $d_p(t)$ is the number of defects found of priority p, in week t. The performance of each strategy on this graph can be explained as an optimisation of the most major defects that can be solved most quickly. The best strategy of the three is arguably Major to Minor, Easy to Hard, with the lowest impact. This is because it ranks defects from the most important and easiest, to the least important and hardest. This metric closely maps the resolution of the most important defects first, which will ultimately lead to less important bugs remaining in the final product, which leads to customer satisfaction. As the issues which are most important and impact the system the most are resolved first, this will boost system reliability.

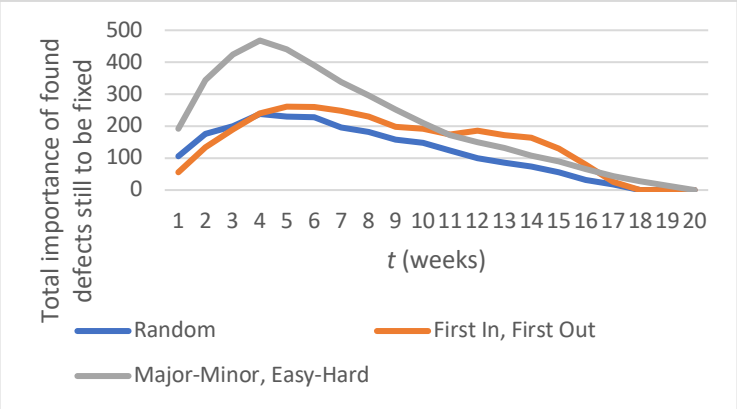

Figure 4: Total importance of found defects to be fixed against t

4.2 Average queue time of major defects still to be fixed

Figure 5 shows the strategy results for the average queue time of major defects still to be fixed. This is calculated as;

$$\frac{df(t) + dr(t)}{df(t)}$$

Where $df(t)$ is the number of defects fixed at week t and $dr(t)$ is the total number of defects remaining unfixed from week 1 to week t. The best strategy is Major to Minor, Easy to Hard, as it prioritises easy defects, which lead to a low metric. Customer satisfaction and system reliability in this metric are closely related, as a strategy the customer is happy with would also promote system reliability. However, all strategies join around week 1, which shows that all approaches would lead to major issues being fixed, but not necessarily with priority.

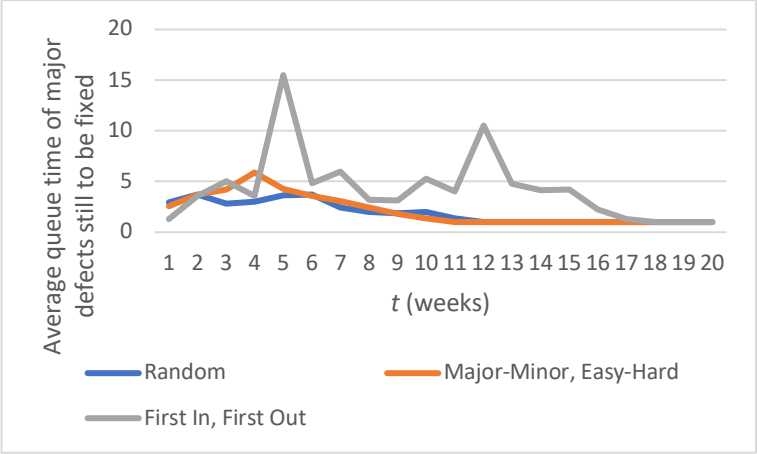

Figure 5: Average queue time of major defects still to be fixed against *t*

4.3 The ratio of defects fixed to defects found

Figure 6 shows the strategy results for the ratio of defects fixed to defects found. This was calculated as;

$$\frac{d(t)}{df(t)}$$

Where *d*(*t*) is the number of defects found in week *t*, and *df*(*t*) is the number of defects fixed in week *t*. As the graph shows, clearly worst strategy to minimise this metric in the beginning is Major to Minor, Easy to Hard. This is because by focusing on major defects which take longer to fix in the early weeks where many defects are being found, few are being fixed which inflates the metric. The other two strategies sit similarly in the middle in the beginning, then improve at around weeks 8-10. Around the same time the Major to Minor, Easy to Hard strategy rapidly improves with the slowing of defect discovery.

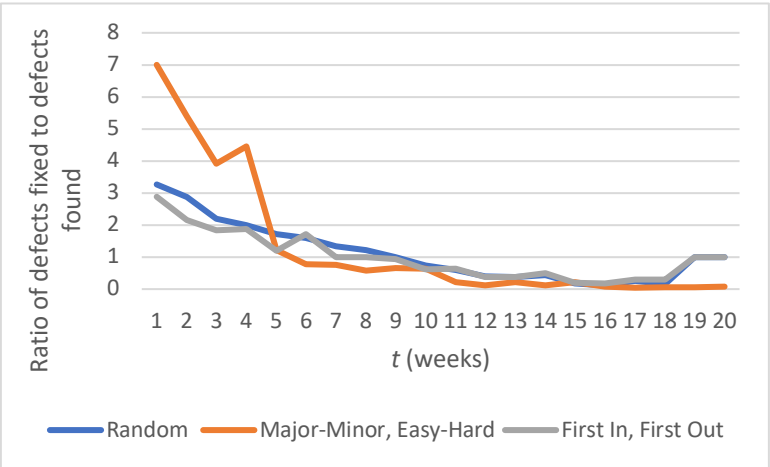

Figure 6: Ratio of defects fixed to defects found against *t*

5. Staff Allocation

To investigate how the metrics change with tester and repairer allocations, three different versions of the Major to Minor, Easy to Hard strategy were developed;

- Version A: Major-Minor Easy-Hard priority
- Version B: $pt = 2$, $pr = 1$ for $1 \leq t \leq 4$, otherwise $pt = 1$, $pr = 2$
- Version C: $pt = 3$, $pr = 0$ for $1 \leq t \leq 4$, otherwise $pt = 1$, $pr = 2$
- Version D: $pt = 2$, $pr = 1$ for $1 \leq t \leq 4$, $pt = 0$, $pr = 3$ for $17 \leq t \leq 20$, otherwise $pt = 1$, $pr = 2$

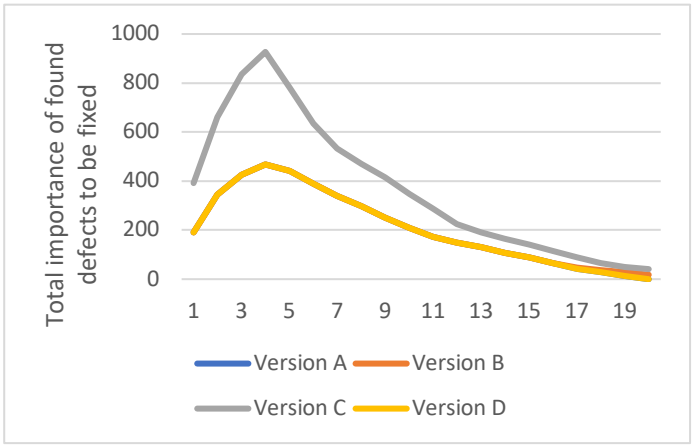

Figure 7: Total importance of found defects to be fixed against *t,* in Versions A-D

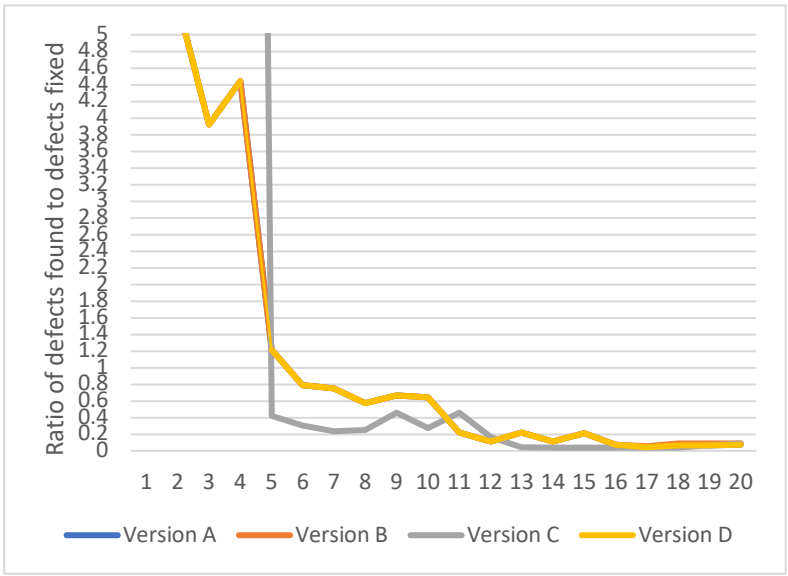

Figure 8: Ratio of found defects to defects fixed against *t,* in Versions A-D

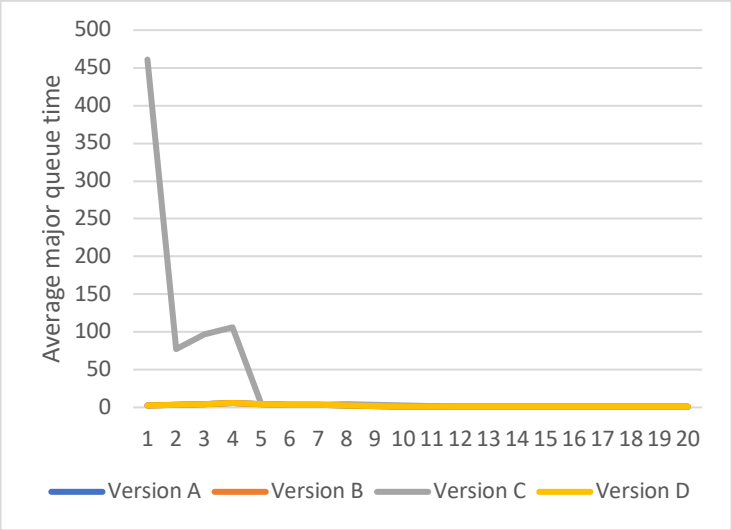

Figure 9: Average queue time of major defects still to be fixed against *t,* in Versions A-D

From the data seen in Figures 7-9, the process of moving testers to repair roles has different effects for different metrics. Version D moves all employees to repair roles from week 17 to week 20. There is no difference in the ratio of defects found to defects fixed as by week 17 almost all defects have been repaired. In version B, there is very poor performance in the ratio of defects found to defects fixed due to finding more defects much faster with this allocation of testers. In the average queue time of major defects, there is a small difference in the total importance of found defects in the versions with more repairers in the beginning weeks.

The version with the most different metrics result was Version C, where there were no repairers in the first four weeks. This allocation of staff resulted in an obviously high ratio, queue time and total importance initially, but after the first four weeks the results became very similar to the other three scenarios.

It is hard to choose an optimum staff allocation from these versions as they all output similar results with the chosen metrics. However, small runs of repair focused or testing focused allocation seem to be beneficial, as shown by Versions C and D. These small runs of repairs and testing would need to be calculated so that enough defects were found to be repaired in a certain number of weeks. More adequate metrics, such as an estimate of number of remaining defects would need to be used to further test this staff allocation.

6. Conclusion

In the simulations, most defects were found and resolved before the 20 weeks testing period had finished. However, the different simulations provided different results in the three metrics. The different simulations of priority and staff allocations had less impact on the defects being resolved in a timelier manner, but rather just the order in which they were resolved. These metrics and simulations are not necessarily independent measures of the health of the software development project.

To maximise system reliability and customer satisfaction, multiple metrics and simulation scenarios, along with other qualitative and quantitative data would be required to make an assessment on the health of the software development.