Software quality management Continual Assessment (2020)

The Pareto principle in Software Engineering

James Mahoney

*MTU, Tralee, Co. Kerry, Eire.* [*james.mahoney@students.ittralee.ie*](mailto:james.mahoney@students.ittralee.ie)

**Abstract**

The Pareto principle also known as the 80/20 rule, states that 80 percent of the consequences come from 20 percent of the causes. This shows an imbalance in the cause and effect. Since this principle has been proven mathematically by Vilfredo Pareto the principle has been observed in many real-world scenarios and has become a driving principle in many different business sectors. This paper looks at the Pareto principle in software development and software testing. The Pareto principle has been observed to hold in software. It is said that 80 percent of the complexity in a code-base comes from just 20 percent of the code and that 20 percent of the features hold 80 percent of the bugs, 80 percent of people will only use 20 percent of the features in an application, and last but not least that the developers developing an application spend 80 percent of their time working on 20 percent of the application.

*Keywords:* pareto principle; Software; SDLC; software testing; web application;

## Introduction

This paper hopes to explain what the Pareto principle is how it works and demonstrate how the Pareto principle is used in the software development life cycle. First It will outline what the Pareto principle is and how it was first noticed and where the Pareto principle has been observed. This paper will look at how the Pareto principle applies to software testing and how methods have been derived from the Pareto principle to approach testing once there is an understanding of the principle then the paper will look at how it is applied to software development and we will look at the pros and cons of these methods.

## Research Method

Research for this paper was performed by viewing different documentation such as papers, journals, books, and websites and bringing the information together to form a good understanding of the topic. There was also a hands-on approach taken by looking at personal applications code repositories and looking at lectures from different modules in I.T Tralee.

## The Pareto principle.

The Pareto principle was first noticed by an Italian economist by name of Vilfredo Pareto in 1906 when he first noticed that in Italy’s land ownership, 80 percent of the land was owned by 20 percent of the people. The principle states that “20% of your time produces 80% of your results, and vice versa”(Garbar, 2020). Once Vilfredo Noticed the phenomenon it was said that he became slightly obsessed with it noticing it everywhere. Even in his garden he noticed that 80percent of the peas came from 20 percent of the pods. (Mar, 2020). Later he created a mathematical formula that would explain the distribution of wealth in Italy. This formula is known as the Pareto distribution. Even though the rule is called the eighty-twenty rule these numbers are used as a rule of thumb and are not necessarily fixed. For example, in 2012, 20 percent of the world's richest countries held 91.62 percent of the wealth of the world (Dunford et al,. 2012).

More examples of the Pareto principle in real-life situations:(80-20 Rule in Software Development, 2020)

* 80% of the products in a manufacturing company are usually on 20% of the line
* 80% of the sick leaves is taken by the 20% of employees
* 80% of usages are on 20% of your files
* 80% of dirt anywhere is on 20% of floor area
* 80% of the time you wash 20% of your clothes
* 80% of the time you wear is 20% of your wardrobe
* 80% of reading happens on 20% of the newspaper i.e. front page, sports page, editorials columns, feature stories
* 80% of your telephone calls will come from 20% of your callers
* 80% of the sales come from 20% of your customers

It has been observed (Applying the 80:20 Rule in Software Development - DZone Agile, 2020) that when it comes to software the Pareto principle can also be observed when looking at the features of an application. It was seen that 45 percent of features of the applications were never used at all,19 percent were rarely used, 16 percent were sometimes used and 20 percent were always used frequently.

## Pareto distribution.

In Paretos's book, he states that there is a law that governs the distribution of income in all countries and, at all times. Explained, if N represents the number of people with wealth larger than a certain income limit x, and A and α are constants, then N=A/x to the power of α, (Dunford et al,. 2012).

log(N) = log(A)−αlog(x)

If the logarithm of the people with a definitive amount is plotted against the logarithm of these incomes it will form a straight line. The line it produces is called the Pareto index. An example of the difference in the Pareto distribution versus normal distribution can be seen below. (Figure 1: Pareto versus normal distributions, 2020)

Chart, line chart

Description automatically generated

Figure 1: Pareto versus normal distributions

## Pareto Principle in software testing

The Pareto principle can also be applied to software and software testing. For software, the Pareto principle would state that a small number of modules would contain the majority of the faults. Studies (Anderson et al., 2007) that have analysed fault distribution have found that the Pareto principle holds true in software also. The study (Anderson et al ., 2007) not only found that a small number of modules contain the majority of the faults but also that even in the modules that contained the least faults that the distribution held true in those modules individually which seems to prove the principle at an even deeper level. Understanding that the Pareto principle is vital for saving time and money testing if a companies software quality team understands this principle the team could put the knowledge to use and determine what the most common bugs are likely to be and where they are most likely to come from. Because the team is aware that the bugs will mostly be coming from these problem areas they can focus resources on these problem areas and therefore save time. Problem areas can be found by looking deeper at the bugs that are showing up. There can be clues such as a lot of bugs are coming when a certain library is called or when certain functionality is being used.

## Pareto Principle in software engineering

Table

Description automatically generatedIn software engineering, the Pareto principle rings true in many other ways. Such as that it takes 80 percent of the time to produce 20 percent of the application or that the users will only use 20 percent of the features of the application, this is leaving 80 percent of the application to elite users. Microsoft themselves noted that when they fixed the top 20 percent of reported bugs this eliminated 80 percent of the bugs and crashes would be eliminated (Kiremire, A.R., 2011). In the paper (Vishal et al ., 2013) the researchers first got the project requirements from the user/client and then asked the user/client to rank each requirement by importance/priority 0 being the highest priority and 9 being the lowest priority. Once collected the developers had the requirements and the set priorities validated by the client. The requirements were then given a level of abstraction 9 being most abstract 0 being least abstract. These values were then added to the table. An example can be seen in Figure 2.

Figure 2: Statements,requirements and abstraction (Vishal

This information allowed the researchers to plot an indifference curve using the priority value and abstraction value and to then choose which requirements were the most important from this graph (Vishal et al., 2013). Once the team had these details the team could then divide the requirements into parts and allocate resources to the requirements.

Chart

Description automatically generated

Figure 3: Indifference curve with prioritised requirements(Vishal et al ., 2013)

The team (Vishal et al ., 2013) concluded that by applying the Pareto principle they improved the output of the team and the model they created way much better than the existing model of rapid applications development.

In software engineering there is also the idea of outsourcing 20 percent of the work, This gives the development team an outsider's perspective. The rule also states that 20 percent of the effort will produce 80 percent of the success. This is done m=by many major companies such as Skye and Microsoft.

Another strategy That has stemmed from the Pareto principle is the minimum viable product. Since according to the Pareto principle 80 percent of the users will only use 20 percent of the functionality the development team can rapidly produce this version of the application with only 2 percent of the features and they can roll this out to see how it is perceived by the users. This minimum viable product once rolled out was developed cheaper and quicker than a full application and the company that produces the application can start making money from the application as soon as possible by only creating 20 percent of the product. If the product is a bust the company has not lost nearly as much money as they would have had they sent the time producing the full application and if it is picked up by the users and is a good product then the application is paying for itself before it is even 20 percent complete. This strategy is a powerful tool for software startups (Tripathi, Oivo, Liukkunen and Markkula, 2019), and a startup not educating itself on the minimum viable product strategy can be the make or break of a software startup. Implementing the minimum viable product strategy allows a small software startup to scale rapidly for all the reasons mentioned previously. This rapid influx of money and rapid user feedback allows a startup to make the changes that the users want quickly and also allows the developers to keep using the 80/20 rule as they apply these changes and add the new features. Projects done in increments like this can be extremely successful when compared to startups that will develop a whole application running over time and budget and if the company fails there is no recourse available.

## Joseph Juran and the Pareto principles

Joseph Juran was a well know evangelist for quality management. He had many inputs to quality management during his lifetime, such as the Juran trilogy(The Juran Trilogy – Continuous Improvement Blog, 2020) which was about the cost of poor quality, his theory was that unless there was a change there would be wastage, it would mean higher costs to change but once the changes were implemented there would be increased revenue. He was also a big believer in the Pareto principle. Even though the Pareto principle was initially about land ownership in Italy he applied it to business. Juran deserves an honorable mention for sparking ideas in Japanese businessmen's minds and igniting a quality control revolution. (Sixsigmadaily.com. 2020).

## Positives of using the Pareto principle in software engineering

There are many positives to using the Pareto principle in software development. For instance, if a team uses the Pareto principle when they are in a requirements stage they can allocate time and resources to the requirements that have the highest priorities. This can reduce the amount of time taken to complete a project and can reduce the budget of the project. When it comes to testing, knowledge of the Pareto principle can guide the software quality department in the testing process this can benefit the project y making sure that the code that will be used most by the software user is tested comprehensively and any areas of code that will be prone to bugs can be isolated and tested thoroughly. This can save time and can money and can be a factor in whether or not the project will fail or not.

## Negatives of using the Pareto principle in software engineering

Usage of the Pareto principle can also have negative effects. If the scoring for requirements is incorrect or changes after the fact and development have already started then the whole process would have to restart again. This can cause problems if there are issues outside the communication of the client/developers. Such an example could be something as common as the stakeholders make a change to a requirement months into the process this could mean that requirements have been allocated solely to something that is only a minor requirement now or that doesn't even exist in the requirements anymore thus amounting to a lot of wasted time and a lot of wasted money. A large challenge of implementing the Pareto principle is that sometimes 80% simply isn't enough and if there is an error in communications within the team it can lead to problems in the development.

## Conclusion

This paper was started with the intent of investigating the Pareto principle and how it related to the software sector. Through research, this paper sought out and explained the Pareto principles beginning where it came from, some of the mathematics behind the principle itself. Some features in software engineering that have been created from the knowledge of the Pareto principle such as a strategy developed to help developers to use the Pareto principle in rapid applications development and went through the steps taken to implement that process. This paper also addresses the idea of a minimum viable product and went through how it can benefit companies if they were to use this in their software development process. It was observed n this paper that the minimum viable product is especially useful for small software startups. This paper also investigated how the Pareto principle can be applied to software testing and that knowledge of the Pareto principle can help to guide software quality management teams in focusing their resources in the areas where most bugs would be found and how the Pareto principle can be used to guide software quality teams which features would rank higher for the end-user. The features that would be used the most will help the team to decide how to rank the code during the testing process. Ranking the code like this can have positive effects on the time spend finding bugs and finding them and fixing the bigger bugs.

## References

Garbar, D., 2020. *Custom Software Testing: How We Use The Pareto Principle*. [online] Belitsoft. Available at:

<https://belitsoft.com/software-testing-services/how-we-use-pareto-principle-and-escape-murphys-law> [Accessed 16 October 2020].

Mar, A., 2020. *13 Examples Of The Pareto Principle*. [online] Simplicable. Available at:

<https://management.simplicable.com/management/new/examples-of-the-pareto-principle> [Accessed 16 October 2020].

Dunford, R., Su, Q. and Tamang, E., 2014. The pareto principle.

ResearchGate. 2020. *Figure 1: Pareto Versus Normal Distributions*. [online] Available at:

<https:/[/www.researchgate.net/figure/Paret](http://www.researchgate.net/figure/Pareto-versus-normal-distributions_fig1_254394933)o[-versus-normal-distributions\_fig1\_254394933>](http://www.researchgate.net/figure/Pareto-versus-normal-distributions_fig1_254394933) [Accessed 16 October 2020].

C. Andersson and P. Runeson, "A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems," in *IEEE Transactions on Software Engineering*, vol. 33, no. 5, pp. 273-286, May 2007, doi: 10.1109/TSE.2007.1005.

Kiremire, A.R., 2011. The application of the pareto principle in software engineering. *Consulted January*, *13*, p.2016.

Vishal, P. and Bhattacharya, S., 2013. Application of the Pareto principle in rapid application development model. Intellisoftwares.com. 2020. *80-20 Rule In Software Development*. [online] Available at:

<https:/[/www.](http://www.intellisoftwares.com/Blogs/80-20-Rule-in-Software-Development)i[ntellisoftwares.com/Blogs/80-20-Rule-in-Software-Development>](http://www.intellisoftwares.com/Blogs/80-20-Rule-in-Software-Development) [Accessed 17 October 2020].

dzone.com. 2020. *Applying The 80:20 Rule In Software Development - Dzone Agile*. [online] Available at:

<https://dzone.com/articles/applying-8020-rule-software> [Accessed 17 October 2020].

Duc, A.N. and Abrahamsson, P., 2016, May. Minimum viable product or multiple facet product? The role of MVP in software startups. In *International Conference on Agile Software Development* (pp. 118-130). Springer, Cham.

Tripathi, N., Oivo, M., Liukkunen, K. and Markkula, J., 2019. Startup ecosystem effect on minimum viable product development in software startups. *Information and Software Technology*, 114, pp.77-91.

Sixsigmadaily.com. 2020. [online] Available at: <https:/[/www.sixsigmadaily.co](http://www.sixsigmadaily.com/remembering-joseph-juran-)m[/remembering-joseph-juran-](http://www.sixsigmadaily.com/remembering-joseph-juran-) quality-improvement/> [Accessed 19 October 2020].

Blogs.mtu.edu. 2020. *The Juran Trilogy – Continuous Improvement Blog*. [online] Available at:

<https://blogs.mtu.edu/improvement/2014/02/27/the-juran-trilogy/> [Accessed 19 October 2020]