Software Quality Management

(Pareto Principle)

Submitted To: Claire Horgan

Submitted By: Colm Gallagher (T00170514)

Computing with Multi Media

Date Submitted: 31/10/2020

**What is the Pareto Principle?**

In 1906, a study carried out by Italian economist, Vilfredo Pareto, found that the wealthiest twenty percent of the country’s population owned eighty percent of the land across his home nation. Pareto surmised that this pattern was reflected across the globe causing a form of social divide. It seems that it was Pareto himself who created the word, “Elite”, in order to describe his theory.

Looking at a case study relating to the world Gross Domestic Product (GDP) just over one hundred years on, in 2011, Pareto’s theory would seem to prove sound. The study finds that over 90 percent of the money across the globe is owned by just twenty percent of the world’s population. (Dunford, Su and Tamang, 2014)

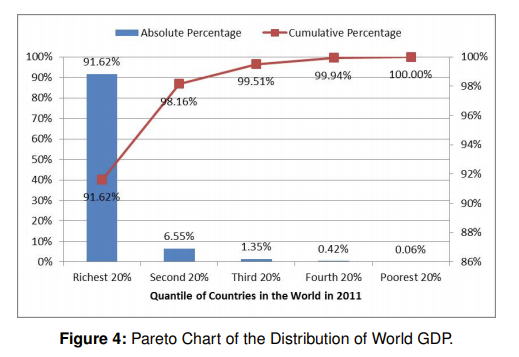


Figure 1 – An image of the Pareto Chart of the Distribution of World GDP - 2011 (Dunford, Su and Tamang, 2014).

Romanian born, American raised, electrical engineer, Dr. Joseph Juran, evolved on this theory in 1941. He later admitted that it was erroneous on his part to give it the name, the *Pareto Principle*, as Pareto had confined his findings only to social and wealth issues.

“’Pareto's principle of unequal distribution applied to distribution of wealth and to distribution of quality losses.’ Although the accompanying text makes clear that Pareto's contributions specialized in the study of wealth, the caption implies that he had generalized the principle of unequal distribution into a universal. This implication is erroneous. The Pareto principle as a universal was not original with Pareto.” (Juran, 1994)

Juran broadened the concept applying it to the wider society and, in particular, the business industry. “*The vital few and the trivial many*” was Juran’s take on what is also commonly called, the *80/20 rule*. It is a simple observation which implies that eighty percent of output is the result of just twenty percent of the input. For example;

* 80% of time allocated to a phone is spent using 20% of the available apps.
* 80% of taxes are paid by 20% of taxpayers. (Project Ricochet, 2020)
* 80% of socialising is done with 20% of close contacts.
* 80% of company results are produced by 20% of company employees. (Rouse, Margaret, 2020)
* 80% of journey to work is travelled using 20% of the available route.
* 80% of software errors are caused by 20% of bugs. (Rooney, 2002)

An excellent example in visual form is provided in this video of a car being drawn in Microsoft Paint. <https://www.youtube.com/watch?v=nXAvgRMmfdw&feature=emb_logo&t=1s>. The *80/20* rule is a guide as opposed to a set rule. The two numbers do *not* need to add up to exactly 100, as also shown earlier with the 2011 world GDP example. (Azad, Kalid, 2020)

These observations have become more prominent in the field of business as companies attempt to capitalise on the findings. If, as appears, the majority of results come from such a relatively small group, it allows said companies to focus more time and resources into the areas in which they deem will produce the greater rewards financially and improve efficiency.

This paper will focus on how the principle can be applied to the field of software development, and the advantages (or disadvantages) of doing so.

**Pareto Principle and Software Development**

The *Pareto Principle* has been applied to a large number of industries over a long period of time. But Dane Balia notes that software engineering, and software developers in particular, are “Still babes in the woods” when it comes to using the practice. (Balia, 2017)

In what could be described as a relatively early example of using the *Pareto Principle* in software development, in 1982, the Federal Reserve Bank of New York tasked McCabe & Associates with carrying out a quality assurance study of their ‘General Purpose Computer Department’ in order to evaluate the efficiency of the payroll system being used at that time. (Schulmeyer, G. Gordon, 2008, pp.127–132)

Termination of a process prior to completion forms (ABEND) played a significant factor in the final analysis brought by McCabe & Associates. It was found that over forty percent of these forms were deemed to be unworthy of even being analysed due to insufficient detail being given about any errors. By implementing the *Pareto Principle* on the remaining ABEND data available, it was calculated that nine error types transpired that year, three of which were accredited to system faults.

Of the other six, human error was to blame. It would appear that the payroll run was, predominantly, a process performed manually which left plenty of scope for a high occurrence of errors. In fact, the results showed that seventy-eight percent of the errors accounted for in the year processed were attributed to human error.

To surmise, the report concluded that the financial implications of such errors were estimated to cost twenty percent of the entire budget of the Federal Reserve’s ‘Central Support Division’.

Thus, it could be suggested that the Pareto Principle details the negative impact on the company in this situation;

* 80% of yearly errors were costing 20% of the Central Support Division’s wages.

As recently as 2002, Microsoft were just one organisation which adhered to the *80/20 rule* with then Chief Executive Officer, Steve Ballmer, stating;

“About 20 percent of the bugs causes 80 percent of all errors, and, this is stunning to me, 1 percent of bugs caused half of all errors.” (Rooney, 2002)

It seems to be an easy logic to understand, with the positive attributes far outweighing the negative ones. To expand on Ballmer’s quote, if eighty percent of the errors are found in just twenty percent of the bugs, it is simple to presume that sixty-four percent of those errors are found in just four percent of the bugs. Then, finally, roughly fifty-one percent of the errors are found in just under one percent of the bugs.

This would suggest a high level of importance being attached to subscribing to the *Pareto Principle* in the computing and information technology industries in the ensuing years since. With so many variables existing in the results of a software development life cycle, it is vital to establish and maintain structural procedures that ensure efficiency, budget control, time management, good customer relations, quality assurance, Etc.

If the software development life cycle process were to be broken down into each step, the *Pareto Principle* can be applied to each phase as follows;

1. **Requirement Gathering and Analysis**

The planning part of the process where it is vital to gather as much information from the customer as possible. What is the end goal? Who is the product aimed at? How many of the requirements are easily achievable?

Once this information is collected, a plan can be initiated to detail what the top priorities for the customer are. Once the lesser needed requirements have been established, there should be a clearer picture of which are “The vital few”, and those which are “The trivial many”.

1. **Design**

Developing both the ‘High-Level Design’ (HLD), and ‘Low-Level Design’ (LLD) documents which will be the basis for the architectural inputs and outputs of the system. As with the requirements phase, the HLD components should outweigh the LLD components in terms of importance, and following the *Pareto Principle*, twenty percent of the input should produce eighty percent of the output.

1. **Implementation and Coding**

One of the more quoted instances of the *Pareto Principle* as shown below.

Approximately 80% of complexity occurs in 20% of code.(Gittens, Yong Kim and Godwin, 2005)

[The first 80% of code is done in 20% of time](http://stackoverflow.com/questions/608748/how-to-avoid-the-80-20-rule-in-software-development)…the remaining 20% of the code takes the other 80% of the time. (Applying the 80:20 Rule in Software Development - DZone Agile, 2020)

1. **Testing**

Another of the often-cited snippets in relation to the *Pareto Principle* within software development. It is explained in detail earlier in the paper with the Microsoft example.

80% of software errors are caused by 20% of bugs. (Rooney, 2002)

Its suggested that prioritising testing to the “Vital few” will save on overall testing time. This is not to discount the “Trivial many”, only to focus on these after the most important features have been tested.

1. **Deployment**

If the preceding phases are executed to a relatively good level of understanding, the deployment phase should find the customer satisfied with the end product. Even if there are changes requested at this point, the likelihood is that these requests would entail working on the “Trivial many” of which many would have hopefully become few.

1. **Maintenance**

80% of changes are made in 20% of the code. (Applying the 80:20 Rule in Software Development - DZone Agile, 2020)

As suggested, only twenty percent of the code would need to be looked at to fix any remaining obstacles leaving the product far more maintainable. This in turn, frees up employees to work on other projects ensuring fluidity throughout the company.

Software managers can implement the uses above to ensure a more efficient workflow across their respective teams which will allow them to funnel their own efforts into delegating tasks to the correct people on their team and concentrating on any risk assessment measures at hand.

Splitting the software development life cycle into the two methods used to date in the software engineering strands, ‘Waterfall’ and ‘Agile’, it can also be established how the *Pareto Principle* effects these methodologies.

**Waterfall**

In one particular study using the ‘Waterfall’ method, the phases were split into tasks. Those that required the most focus, and those which could be disregarded. What began as a list of 144 tasks soon became just 85 left to be completed. This saved time and resources, therefore giving improved efficiency.

“The results showed that reducing the number of tasks by half (to 85 tasks), they could still maintain 70%-80% of the original productivity.” (Kiremire, Ankunda R., n.d., 2011)

**Agile**

It is feasible that the ‘Agile’ method could be construed as to be the most suited method to the *Pareto Principle,* due to its flexibility. By its nature, ‘Agile’ is about adapting to challenges as a process develops and so being able to focus eighty percent onto the most pressing of issues lends itself to appearing to be a befitting match for the *Pareto Principle.* (Appling 80/20 Rule (Pareto Principle) in Software Development | Xicom, n.d.)

The Pareto Principle isn’t perfect by any means. It’s important to state that it is not a law, but a guide. So, it does have disadvantages. For example, eighty percent of users may only use twenty percent of an application. Eighty percent of employees may produce only twenty percent of output.

**Pareto Principle – the future**

In “Goodbye Pareto Principle, Hello Long Tail:” (Brynjolfsson, Hu and Simester, 2011), the authors hypothesise whether the *Pareto Principle* has become a less useful observation since the dawn of the internet. Their study would suggest that more products are available than ever with online markets becoming so common and that the *80/20 rule* is not as reliable as it once was.

“On the Internet, the Pareto principle may be giving way to the “long tail.”

There is also the suggestion that Artificial Intelligence will skew the 80/20 ratio of the Pareto Principle in future generations with what is termed as ‘Super-Paretos’;

“The dirty little productivity secret of big data is that Pareto’s 80/20 insight has decayed into empirical anachronism. Analytically aggressive firms increasingly see Pareto proportions closer to 10/90, 5/50, 2/30, and 1/25. Pareto’s “vital few” becomes a “vital fewer.”” (Schrage, 2017)

**Pareto Principle in conclusion**

A fascinating personal take that I have on the Pareto Principle is that I can use it to improve my efficiency when it comes to the matter of writing reports and focussing better on my time management in general.

*The 96-minute rule* is an offshoot of the 80/20 rule which I plan on introducing to my work schedule. Basically, if you take a typical eight hour working day, the total minutes totals 480. Twenty percent of that equals 96 minutes. The theory is that if we focus our full attention for that period of time, with no distractions such as mobile phones or coffee breaks, we will produce eighty percent of our workload from just that twenty percent window of our time. It would appear to be a rewarding and productive rule to follow. (The 96-minute rule, and other timely tips, n.d.)

It is hoped that this paper has displayed a grasp of what the *Pareto Principle* is, and how it benefits the world of software development in particular.

The *Pareto Principle* is still as relevant today as it was in 1906 which can be proven with this topical statistic;

Twenty percent of ‘Superspreaders’ would appear to be responsible for eighty percent of local Coronavirus transmission. (Duszyński, 2019)

**References**

Anon 2020. *Applying the 80:20 Rule in Software Development - DZone Agile*. [online] dzone.com. Available at: <https://dzone.com/articles/applying-8020-rule-software> [Accessed 30 Oct. 2020].

Anon 2020. *Software development, the Pareto Principle, and the 80% solution | Project Ricochet*. [online] Available at: <https://projectricochet.com/blog/software-development-pareto-principle-and-80-solution> [Accessed 30 Oct. 2020].

Anon n.d. Appling 80/20 Rule (Pareto Principle) in Software Development | Xicom. Available at: <https://www.xicom.ae/blog/pareto-principle-in-software-development/> [Accessed 1 Nov. 2020].

Anon n.d. The 96-minute rule, and other timely tips. [online] Available at: <https://www.theglobeandmail.com/report-on-business/careers/management/the-96-minute-rule-and-other-timely-tips/article13816023/> [Accessed 1 Nov. 2020].

Azad, Kalid, 2020. *Understanding the Pareto Principle (The 80/20 Rule) – BetterExplained*. [online] Available at: <https://betterexplained.com/articles/understanding-the-pareto-principle-the-8020-rule/> [Accessed 29 Oct. 2020].

Balia, D., 2017. *The Pareto Principle Applied To Software Development*. [online] Medium. Available at: <https://medium.com/360-developer/the-pareto-principle-applied-to-software-development-b6c5f4d9d863> [Accessed 30 Oct. 2020].

Brynjolfsson, E., Hu, Y. (Jeffrey) and Simester, D., 2011. Goodbye Pareto Principle, Hello Long Tail: The Effect of Search Costs on the Concentration of Product Sales. *Management Science*, 57(8), pp.1373–1386.

Dunford, R., Su, Q. and Tamang, E., 2014. The Pareto Principle. [online] Available at: <https://pearl.plymouth.ac.uk/handle/10026.1/14054> [Accessed 30 Oct. 2020].

Duszyński, M., 2019. *Pareto Principle & the 80/20 Rule (Updated for 2020)*. [online] ResumeLab. Available at: <https://resumelab.com/career-advice/pareto-principle> [Accessed 31 Oct. 2020].

Gittens, M., Yong Kim and Godwin, D., 2005. The vital few versus the trivial many: examining the Pareto principle for software. In: *29th Annual International Computer Software and Applications Conference (COMPSAC’05)*. 29th Annual International Computer Software and Applications Conference (COMPSAC’05). pp.179-185 Vol. 2.

Juran, J.M., 1994. *The Non-Pareto Principle; Mea Culpa*. [online] Available at: </paper/The-Non-Pareto-Principle%3B-Mea-Culpa-Juran/587b2c46da9fd6f63d50d992bdb70dabbad39407> [Accessed 31 Oct. 2020].

Kiremire, Ankunda R., n.d. THE APPLICATION OF THE PARETO PRINCIPLE IN SOFTWARE ENGINEERING. *2011*, p.12.

Rooney, P., 2002. *Microsoft’s CEO: 80-20 Rule Applies To Bugs, Not Just Features*. [online] CRN. Available at: <https://www.crn.com/news/security/18821726/microsofts-ceo-80-20-rule-applies-to-bugs-not-just-features.htm> [Accessed 31 Oct. 2020].

Rouse, Margaret, 2020. *What is Pareto principle ? - Definition from WhatIs.com*. [online] WhatIs.com. Available at: <https://whatis.techtarget.com/definition/Pareto-principle> [Accessed 29 Oct. 2020].

Schrage, M., 2017. AI Is Going to Change the 80/20 Rule. *Harvard Business Review*. [online] 28 Feb. Available at: <https://hbr.org/2017/02/ai-is-going-to-change-the-8020-rule> [Accessed 1 Nov. 2020].

Schulmeyer, G. Gordon, 2008. Handbook of Software Quality Assurance (4th edition). In: *Handbook of Software Quality Assurance (4th edition)*, 4th ed. Artech House.pp.121–148.