Improvement of the Software Development Process to Achieve Higher Quality

William Rivera García
Master of Engineering Management Program
Dr. Héctor J. Cruzado
Graduate School
Polytechnic University of Puerto Rico

Abstract *— This article seeks to explore how to take an existing software developing process in the Avionics Department of an Air Force base and improve it so that the product quality is increased and lower the number of defects found by the customer by 30%. Extensive analysis of past development was performed accompanied by Root Cause Analysis to determine the most repeating causes. The results suggest that functional defects were the most common with causes such as lack of experience in the team, no coding standard, lack of communication with the client, ambiguous requirements, among others. In order to achieve the objective, a coding standard was created, routine meetings with the clients and pair programming was established. The results showed a good effect in the reduction of defects found by the client. These results suggest that tackling these common causes will greatly help the development team to deliver a higher quality product and thus improving their process.*

Key Terms *— Defects, Development Process, Root Cause Analysis, Software Quality.*

INTRODUCTION

Software quality is the core of any software development company. It is very important for teams to do their projects correct the first time because it brings better client satisfaction and performance. An effective team can deliver software in a timely manner with little to no rework. As the world continues moving forward into the future of Software Development, quality is the focus of many companies and departments. Being able to constantly produce quality software to the client with the least number of defects possible is the goal of every software developer and thus is the main motivation for this project. This paper explores how an existing software developing process in an Air Force base, Department of Avionics, could be improved to ensure a higher quality deliverable. Predictability is driven by software quality. Do it once and do it right, and less re-work, less efficiency volatility and overall improved results can occur. Things are shipped on schedule, and more productively they are installed. Bad quality is much harder to handle. The purpose of this project is to lower the number of defects identified by the client in the deliverables by a minimum of 30%.

LITERATURE REVIEW

Software Quality is a concept that has been changing as time passes in an effort to achieve maximum effectiveness while producing the right product. A big aspect of this is for developers to be able to produce a product with little to no rework needed after being delivered to the client. An article exposes how effective communication from the client affects the software development process, with three case studies performed which provide ample data supporting the idea that less communication leads to higher defects [1].

Traditional software development practices and methods have been proven to not be efficient enough to mitigate the fast pace of real-world problems like requirement changes which is why Agile methodology are being widely adopted to increase productivity and quality [2]. A study looked into the effects of teamwork quality as it pertains to project success in software development teams, which is perceived to have a small to large effect on team performance [3]. A study draws a systematic relationship between the policies for setting deadlines and the quality of software products, especially when teams want to focus on quality but fear the negative consequences of missing a deadline [4]. Many firms are investing in

product quality due to the exponential growth of the software industry in order to remain competitive by delivering high quality software on time [5].

ANALYSIS

This section is composed of three parts: Defect Classification, Identifying Worst Offender and Root Cause Analysis.

Defect Classification

Historical data was gathered from various past developments in order to classify the defects reported by the client into different categories. Table 1 shows data gathered from the beginning of year 2020 with the amounts of Source lines of Code and the reported defects. Figure 1 shows data analysis performed on the defects with a P chart which shows an average amount of defects of 39%.

Table 1
Historical Data from Department 2020

Date (2020)	SLOC (Source Lines of Code)/Pages of Documentation	Reported Defects	Ratio
January	628	254	0.40
February	584	217	0.37
March	379	185	0.49
April	395	175	0.44
May	585	212	0.36
June	466	169	0.36
July	635	226	0.36
August	570	222	0.39

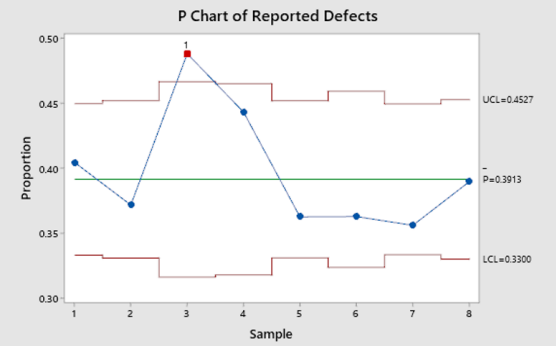

Figure 1
P Chart of Reported Defects

Identifying Worst Offender

Defects were divided into categories with the purpose of determining the defects that occurs more regularly during development. The categories defined by this exercise were Functional, Cosmetic and Off-Suite defects. Table 2 presents these categories with their description.

Table 2
Defect Categorization

Functional	Defects where a requirement wasn't properly coded in the package or didn't comply with customer's needs.
Cosmetic	Documentation mistakes like typos, typos in the comments of the code, duplicated or copy-paste errors.
Off-Suite	Defects that got carried over from previous Suites.

Further analysis on this data showed that the majority of reported defects from the client were Functional defects, followed by Cosmetic then finally Off-Suite. Figure 2 below shows this relation in the form of a Pareto Chart. Functional defects being the worst offenders, were selected to be the focus of this project.

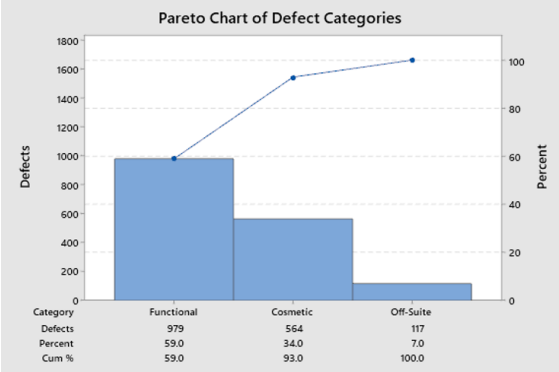

Figure 2
Pareto Chart of Defect Categories

Root Cause Analysis

Cause and effect analysis and other problem-solving tools taken and implemented from Scaled Agile framework. These took the form of a fishbone diagram which was used to identify the possible causes to the defects. The possible causes were divided into four categories: People, Process, Tools and Management. Some of the causes found were lack of experience, not having a coding standard, lack of communication with the client, ambiguous requirements, having to use and old programming language, no existing development tools, rigorous deadlines.

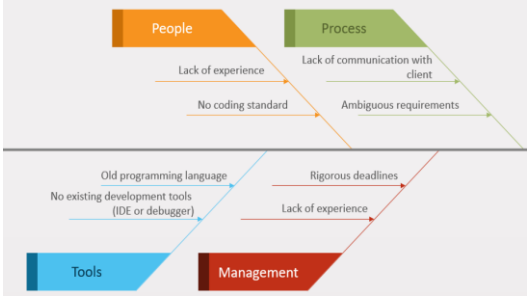

Figure 3
Fishbone (Ishikawa Diagram)

RESULTS

The Defect Classification showed a large number of reported defects by the client. These defects were successfully collected and categorized into three main categories: Functional, Cosmetic and Off-Suite. Data analysis showed that the worst offender were Functional defects by a large margin of 59%, followed by Cosmetic defects with a 34% and lastly Off-Suite defects with a 7%. Root Cause Analysis showed various causes in different areas that were present more than once. Lack of experience from the management and the development team was one of these causes.

CONCLUSION

The implementation of a coding standard to aid in the development of software for new developers and to maintain a standard across the project shows to be the best solution. Along continued communication with client, pair programming, training sessions and peer reviews. The major causes identified in the analysis were the lack of experience from developers, the lack of input from the client, ambiguous requirements among others which lead to a high number of defects. Future work for this project would be to motivate management to follow through and implement the changes necessary and tackle the other root causes that weren't covered by the scope of this project.

REFERENCES

[1] Korkala, Mikko & Abrahamsson, Pekka & Kyllönen, Pekka. (2006). A Case Study on the Impact of Customer Communication on Defects in Agile Software Development. Proceedings - AGILE Conference, 2006. 2006. 76-88. 10.1109/AGILE.2006.1.

[2] Ahmed, S. Ahmad, N. Ehsan, E. Mirza and S. Z. Sarwar, "Agile software development: Impact on productivity and quality," 2010 IEEE International Conference on Management of Innovation & Technology, Singapore, 2010, pp. 287-291, doi: 10.1109/ICMIT.2010.5492703.

[3] Lindsjørn, Yngve & Sjøberg, Dag & Dingsøyr, Torgeir & Bergersen, Gunnar & Dybå, Tore. (2016). Teamwork Quality and Project Success in Software Development: A Survey of Agile Development Teams. Journal of Systems and Software. 122. 10.1016/j.jss.2016.09.028.

[4] Austin, Robert. (2001). The Effects of Time Pressure on Quality in Software Development: An Agency Model. Information Systems Research. 12. 195-207. 10.1287/isre.12.2.195.9699.

[5] Slaughter, Sandra & Harter, Donald & Krishnan, M.. (1998). Evaluating the Cost of Software Quality. Commun. ACM. 41. 67-73. 10.1145/280324.280335.