New approach of the Customer Defects per Lines of Code metric in Automotive SW Development applications

I-A Sandu¹, A Salceanu¹, O Bejenaru¹
¹Gheorghe Asachi” Technical University of Iasi, Faculty of Electrical Engineering
E-mail: asalcean@tuiasi.ro

Abstract. As the market of the SW based electronic components in the automotive industry is increasing with high speed, both Original Equipment Manufacturers (OEM) and suppliers need to find solutions for delivering quality products which fulfil customer requirements and safety regulations. One measure of quality is represented by the number of Customer Defects. But how to normalize the Number of Customer Defects metric in order to measure and compare the results of this metric for any kind of automotive SW development project, independent of the complexity and code size? We experienced one possible solution by measuring the metric Customer Defects per Lines of Code on several projects. We analysed if and how code size and implicitly complexity influence the quality of the Automotive SW Development applications. Are complex systems more prone to errors? Should we invest in quality for low complexity systems? These are the main questions we tried to answer.

1. Necessity of a relevant metric
In order to fulfil the safety and quality standards and to keep the pace with the market, automotive companies are investing in quality strategies for the delivered software based electronic components. Increasing the faults detection rate within the development phases will reduce problem fixing effort and the test effort. More precisely, detection of 10% more defects in software design or coding phases can lead to a potential saving of 3% of the total product development cost [1]. The error correction cost can even increase up to 90 times in post-production phase compared to concept phase [2]. Price of recalls comprises beside fault fixing costs, also legal costs and image costs.

How do code size and implicitly complexity of an automotive SW Project influence the quality? One dimension of the code size is represented by the number of lines of code (Source Lines of Code [3]). SLOC is used to determine the size and cost of the software development projects. Considering this metric, we can compare the size and complexity of different projects, as presented in equation (1).

\[ \text{Source lines of code} = \text{Total project lines of code} - \text{Number of empty lines and commented lines} \] (1)

When we consider a supplier, quality can also be measured through the number of problems discovered by the customer (Original Equipment Manufacturer) for the delivered software product. But how to normalize the Number of Customer Defects metric in order to measure and compare the results of this metric for any kind of automotive SW development project, independent of the complexity and code size?

2. Certified achievements of the domain
One practice is to represent the number of external customer defects reported per 1000 lines of code, as presented in equation (2):

\[
\text{Customer Defects per Lines of Code (CDLC)} = \frac{\text{Number of customer defects}}{\text{(kilo lines of code)}}
\]

The generic software industry (not specific to automotive) average is about 15 – 50 errors per 1000 lines of delivered code. This is known as the defects per kLOC (1000 lines of code). In Microsoft Applications the average for CDLC is about 10 – 20 defects per 1000 lines of code during in-house testing and 0.5 defects per kLOC in production [4]. CDLC is also known as Defect Density metric [5].

3. Description of the method

Quality of an automotive software development project can be measured by tracking the number of defects or problem reports. Defects can be internal (discovered during the development process, before releasing the product to the customer) or external (discovered by the customer) [6].

On the organization level, we consider that the complexity of a software development project is proportional with the number of lines of code. Based on a reference project with lowest number of lines of code, it can be evaluated the complexity of any other project.

In order to determine if there is any connection between quality and code size / complexity in the automotive software development projects, equation (2) can be used for measuring the Customer Defects per Lines of Code. This allows comparing the results of projects having different complexities with regard to the total number of lines of code. Equation (2) can be applied to projects using different development methodologies like V-Cycle or Agile [7].

4. Obtained results

We applied formula (2) for calculating CDLC on 7 projects with a range of complexity between 100 and 4000 kilolines of code. We measured the results on monthly basis on a timeframe of 32 months. The projects in scope were using V-Cycle development methodology [8]. In order to calculate CDLC every month we collected separately the number of customer defects and the number of lines of code for each project. For the total number of lines of code we considered all the modules and code categories that were integrated in the end product: new created code, software modules / components from suppliers, Open Source software, re-used code and legacy code. There were excluded from the calculation only empty lines and comments. After we obtained the total number of lines of code, we transformed this into kilo lines of code (kLOC) by applying equation (3).

\[
kLOC = \frac{\text{Total number of lines of code}}{1000}
\]

\[\text{Figure 1: Customer Defects per Lines of Code at the End of the Projects}\]
In figure 1 we present the values of CDLC, the number of kilo lines of code and the number of customer defects at the end of the project. We can identify that project 1 although it has the highest number of lines of code; the number of CDLC is lower than other projects which are less complex: project 3, project 4 and project 7. Projects 2, 5 and 6 have lower values in comparison to project 1 for both CDLC and lines of code.

A high value for the metric Customer Defects per Lines of Code should indicate that quality is affected. Project 1 which is 2 times more complex than project 7 in regard to the number of lines of code, has actually about 30% less customer defects. The same situation appears when we compare project 3 and project 2: although project 3 has less LOC than project 2, it has an increased number of customer defects.

CDLC distribution for each project is presented in figure 2. Based on our research, we can identify that the value of CDLC is increasing as we reach the end of the project. This can be explained by the fact that the deliveries to the customer are containing the majority of the features at the end of the project. More than this, the full system is also available at that point in time. For 4 projects out of 7 the final CDLC values are within 1 and 3 interval. Project 6 has the lowest CDLC values overtime. By analysing figure 1 we identify that this is a low complexity project which it has also a low number of customer defects. Project 7 has the highest values of CDLC over time, although it has an average complexity.

For project 5 we can identify spikes and even a descending trend over time starting with month 28. The number of Customer Defects and number of lines of code are cumulative numbers over time. The ascending trend of CDLC at the beginning and the descending trend of CDLC at the end of the project could be explained by a low income rate of customer defects by end of the project. We can identify in figure 2 that starting with month 15 the project team took corrective actions in order to reduce the number of incoming customer defects. By reaching at the end of the project one of the lowest values of CDLC from all the 7 projects analysed, project 5 has a medium complexity with regard to the number of lines of code.

By analysing the results of CDLC metric measured on a period of 32 months on 7 automotive SW development projects, we concluded that code size / complexity does not influence the quality of the
end product. For complex projects, a high number of kLOC does not imply also a high number of customer defects, the same as for projects with low number of lines of code do not apply a low rate of CDLC. This means that in both simple and very complex automotive SW development applications, built-in quality and testing activities are very important. The efficiency of these phases can be monitored by using the Phase Containment Metric [9]. Also very important is the collaboration with the customer from incipient stages until the testing and validation phases. Testing and validation of the end product can be done by joint activities involving both parties: supplier and customer.

5. Conclusions and further attempts

CDLC is a very powerful metric which can be presented over time or at the end of a project, as we proceed in this paper. Organizations should establish targets for this metric and projects should take corrective actions and measures in order to fulfil these targets. Any deviation from the constant trend of the metric should also trigger immediate root cause analysis and continuous product quality and process improvement. CDLC can be monitored on monthly basis or even for each delivery to the customer. Being normalized, CDLC can be measured on project level or for all projects from an organization. The CDLC values can be compared, and could be used for Lessons Learned for future projects when measuring at the end of the projects. CDLC can also be used as immediate improvement action indicator when targets are set and the metric is measured within the project lifecycle.

As further development and refinement of this metric, fault severity classification (major, minor) needs to be defined either on project or on organizational level. In order to determine what needs to be filtered out from measurement, it can be used as starting point the business goal(s) and the questions this metric should answer. In the metric definition should be documented which type of defects will be counted (e.g.: all, only major ones, only the customer visible, etc.).

References

[1] Ebert C, Dumke R, 2007, Software Measurement, Springer, Heidelberg, New York, pp 245-300
[2] Seidler D, Southworth T. (2016, November 21). IBM Rational Automotive Engineering Symposium 2013, Source - Herstellerinitiative Software (Audi, BMW, Daimler, Porsche and Volkswagen). [Online]. Available: https://www.ibm.com
[3] Nguyen V, Deeds-Rubin S, Tan T, Boehm B, A SLOC Counting Standard, Center for Systems and Software Engineering, 2007, University of Southern California
[4] McConnell S C, Code Complete's 2nd edition [Online]. Available: http://www.cc2e.com. [Accessed 22 Nov. 2017]
[5] Nagappan N, Ball T, Use of relative code churn measures to predict system defect density, Proc. of the 27th Int. Conf. on Software engineering, May 15-21, 2005, St. Louis, MO, USA
[6] Hallowell D L, Six Sigma Software Metrics Maturity, Part 1, Six Sigma Advantage Inc, 2003 [Online]. Available: https://6sigma.com
[7] Beck K, Beedle M, Bennekom A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin R, Mellor S, Schwaber K, Sutherland J and Thomas D. (2001). Manifesto for Agile Software Development. [Online]. Available at: http://www.agilemanifesto.org/ [Accessed 03 Apr. 2017]
[8] Royce W, Managing the Development of Large Software Systems, 1970, Proc. of IEEE Symp. WESCON, 26 (August) pp 1–9
[9] Sandu I A, Salceanu A, Metrics Improvement for Phase Containment Effectiveness in Automotive Software Development Process, 2017, Proc. of the 10-th International Symp. on Advanced Topics In Electrical Engineering (ATEE 2017), 23-25 March, Bucharest, Romania pp 661-666
[10] Sandu I A, Salceanu A, Applications of the Phase Containment Effectiveness Metric in Automotive Industry Agile SW Development, 2017 Proc. of the 22nd IMEKO – TC4 International Symp. 14-16 September, Iasi, Romania pp 45-49.