**ACTA IMEKO**
**ISSN: 2221-870X**
*December 2018, Volume 7, Number 4, 3-8*

# New approach to agile cycles containment effectiveness metrics in automotive software development

**Ionut-Andrei Sandu[1], Alexandru Salceanu[1]**

[1] *"Gheorghe Asachi" Technical University of Iasi, Mangeron Str. 67, 700050 Iasi, Romania*

ABSTRACT
In an ideal agile development team, defects should not exist. However, in reality, especially in automotive agile software development, there must be a mechanism for handling defects and tracking them to closure. In this article, we describe the benefits of and principles underlying the measurement of defects handling metrics in automotive programs and in organizations that have adopted agile software development. We present the Iteration Containment Effectiveness, Program Increment Containment Effectiveness, and Defect Debt Trend metrics. The advantages acquired thereby are demonstrated by a detailed example of a real application concerning how to measure the classic Phase Containment Effectiveness metric on the Iteration (Sprint) and Program Increment (Scum of Scrums / Scaled Agile) Level.

**Section:** RESEARCH PAPER

**Keywords:** Iteration Containment Effectiveness; Program Increment Containment Effectiveness; Defect Debt Trend; Phase Containment Effectiveness; software quality metric; agile; defects handling

**Citation:** Ionut-Andrei Sandu, Alexandru Salceanu, New approach to agile cycles containment effectiveness metrics in automotive software development, Acta IMEKO, vol. 7, no. 4, article 2, December 2018, identifier: IMEKO-ACTA-07 (2018)-04-02

**Editor:** Vilmos Pálfi, Budapest University of Technology and Economics, Hungary

**Received** March 13, 2018; **In final form** October 2, 2018; **Published** December 2018



**Funding:** This work was supported by "Gheorghe Asachi" Technical University of Iasi, Faculty of Electrical Engineering

**Corresponding author:** Ionut-Andrei Sandu, e-mail: siandrei@gmail.com

## 1. THE IMPORTANCE OF MEASURING THE PHASE CONTAINMENT EFFECTIVENESS METRIC IN AGILE SOFTWARE DEVELOPMENT

In the automotive industry, agile software development began to be used among developers as far back as 12 years ago, according to Kugler Maag Cie"s study in 2015 [1]. Since then, more and more automotive companies (OEMs and suppliers) that develop software-based electronic components are implementing the agile methodology. This is mainly because companies must keep pace and be flexible with constantly changing requirements, especially in current times, when the time to market is decreasing.

Organizations adopting the agile methodology also implicitly implement continuous process improvement, as teams and organizations must be effective and efficient. Agile process transformation also implicitly triggers improvement actions and measures for software development processes. In this way, projects and organizations support and successfully fulfill the requirements of the process assessment models (e.g., Automotive Software Process Improvement and Capability Determination, A-SPICE® [2]).

One of the principles of the Agile manifesto is "working software is the primary measure of progress [3]." The ideal situation is that deliveries have no faults that affect the end user or faults that are introduced due to the incorrect implementation of the requirements.

For the minimization of the defects rate, the classic software development approach is to use the Phase Containment Effectiveness (PCE) metric to calculate the effectiveness of the verification in each of the development phases. The PCE metric answers the following questions: How efficient is the verification process? Which phases escaped defects? Which phases found/did not find those defects?

How can the above questions be answered concerning programs and organizations that have adopted agile software development?

We propose here a method of how to apply the PCE metric to organizations and teams that have developed software for the automotive industry using the agile software development approach.

## 2. PREVIOUS ACHIEVEMENTS IN THIS FIELD

PCE was introduced in 1997 as a software quality improvement metric by A. R. Hevner [4]. In 2003, it was also adopted in the Six Sigma (a disciplined, data-driven methodology for eliminating defects) [5]. This metric provides the ability to measure verification (a review, inspection, or unit-testing software method ensuring that each software unit satisfies its design) effectiveness and allows the software development team to improve their software development process.

The PCE metric can also be used for measurement in automotive software development by applying it in the specific software development and test phases [6].

Faults can be classified as either errors or defects depending on the phase into which they were injected and the phase in which they were found.

Errors are faults that are discovered in the proper phase into which they were injected (e.g., design faults caught by design reviews). Defects are faults that were not identified in the development phase (e.g., design faults caught in code reviews or software tests).

Ideally, all faults should be discovered in the phase in which they were introduced, leading to an idealistic PCE of 100%. Considering that in the automotive industry the rate of software-related recalls increased from 5 % in 2011 to 15 % in 2015 (Stout Risius Ross"s study, based on data from the United States National Highway Traffic Safety Administration [7]), improved phase containment is needed in automotive software development.

Increasing the faults detection rate within the development phases will reduce the problem resolution 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 % in the total product development cost [8].

The error correction cost can even increase up to 90 times in the post-production phase compared to the concept phase [9]. The price of recalls comprises, besides fault-fixing costs, legal costs and image costs. At present, researchers have not decided how to apply the PCE metric in agile software development.

## 3. OUR SPECIFIC APPROACH

Agile software development is executed in iterations (according to Scaled Agile Framework SAFe® model [10] or sprints in SCRUM [11]). Working software (ideally fault-free) should be delivered at the end of every iteration. Because incremental build and continuous integration is undertaken in agile software development [12], current delivery is used in the next iteration to add features on top. However, if a delivery containing undiscovered faults from Iteration *N* is used to add new functionalities for the upcoming deliveries >*N*, the undiscovered faults are also implicitly translated to these deliveries.

Because defects can escape from one iteration to another in agile software development, the iteration itself can be considered as a phase in the classic PCE metric. Faults that have escaped identification from one iteration to another (i.e., those that are inherited by the next iteration) can be monitored and reduced by analyzing and taking appropriate actions when measuring the PCE for iterations, which we call Iteration Containment Effectiveness (ICE):

- Iteration errors: faults caused during iteration N and discovered/solved during Iteration N (e.g., during architecture review, code review, software testing).
- Iteration defects: faults caused during Iteration N and detected/solved during Iteration >*N* (the upcoming iterations) or by the customer

The total number of faults is obtained by equation (1):

$$Iteration\ Faults = Iteration\ Errors + Iteration\ Defects \quad (1)$$

ICE can be calculated for each iteration by calculating equation (2):

$$ICE\ for\ Iteration\ N = \frac{Iteration\ Errors}{Iteration\ Faults} \cdot 100\% \quad (2)$$

How can defects that will be fixed in the next iterations be handled? A Problem Report (PR) should be opened, the defect should be added to the Program Backlog, and it should be prioritized accordingly. If the Defect Debt (the number of open PRs) is increasing, a special iteration might be required for fixing bugs.

How can defects (issues caused from previous iterations) that have been discovered and will be fixed in the current iteration be handled? If a defect that is not related to the current iteration but is severe enough that an immediate fix is required, a PR should be created. This will ensure transparency and alignment with all the stakeholders, as the scope of the current iteration may be affected. It will also support the project team concerning future situations in which similar problems may be detected. If the user story is closed, it is advised that a fix task or a PR is created anyway and linked with the initial story. [13]

How can errors that are discovered and fixed in the same iteration be handled? Valuable data would be lost if issues found during testing activities are not captured. However, entering and administrating formal PR entries is probably an overhead. This issue can be handled in either of the following ways:

1) Such errors should be tracked to closure in a separate list
2) The current workflow used by the team should be followed. For example, when the development of a current user story is complete, the user story should be transitioned to a verifying state. If it did not pass the verification, the ticket should be returned to the implementing state. Only after the developer fixes the discovered error is the user story transitioned again to the verifying state.

## 4. SYNTHESIS OF OBTAINED RESULTS

In the following example, we outline a Program Increment with five execution iterations. Iteration 15 is the last one, executed on the creation date of the report.

Equations (1) and (2) allow us to calculate the ICE for each Iteration. This is how the values in Table 2 were obtained for a specific program.

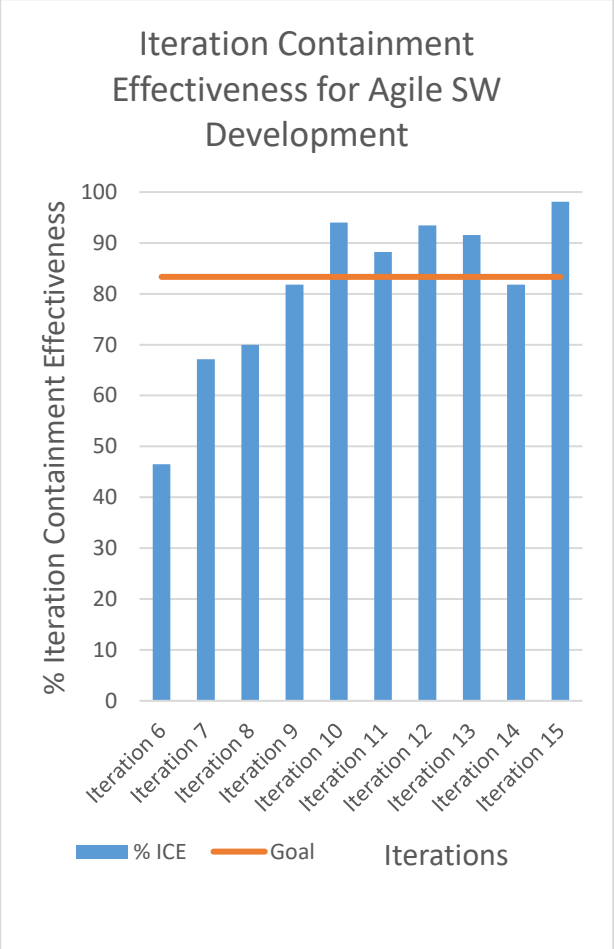


Figure 1. ICE values for a specific Program using Agile.

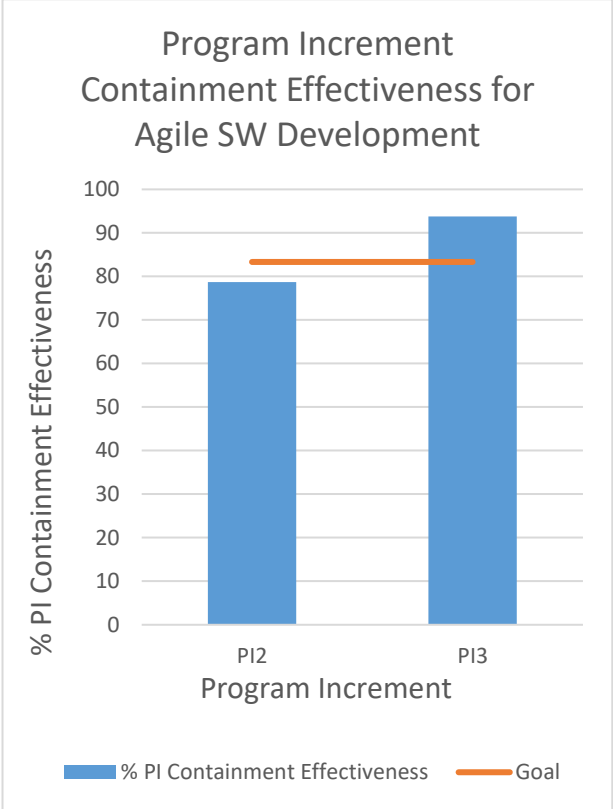


Figure 2. Program Increment Containment Effectiveness values for a specific Program using Agile.

Furthermore, in agile software development, the classic PCE metric can be applied at the iteration level. To understand an iteration in whose development and testing phases defects were not identified, the classic PCE for Development metric [5] (based on the development and test phases used in the iteration) and the classic PCE for Testing metric [5] (considering only the development and test phases used in the iteration) should be used at the iteration level.

As a Program Increment (PI) consists of several iterations and as the unit of a program execution is represented by the PI [10], ICE can also be applied to the PI level by applying the same mechanism. We therefore propose Program Increment Containment Effectiveness (PICE) in equation (3), which involves

- PI errors: faults discovered during *PI N*
- PI defects: faults that escaped from *PI N* and were detected during *PI* > *N* (the upcoming PIs) or by the customer

$$Program\ Increment\ N\ Containment\ Effectiveness = \frac{\text{Program Increment Errors}}{\text{Program Increment Errors+Program Increment Defects}} \cdot 100\% \quad (3)$$

The ICE and PICE metrics can also be analyzed together with the trend of open defects for each iteration/*PI*. In short,

Table 1. ICE values for a specific program using the agile methodology.

| | | It.6 | It.7 | It.8 | It.9 | It. 10 | It. 11 | It. 12 | It. 13 | It. 14 | It. 15 | Customer | Total errors | Total defects | Total faults | % ICE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Program increment 2 | Iteration 6 | 20 | 3 | 0 | 0 | 1 | 10 | 2 | 0 | 0 | 0 | 7 | 20 | 23 | 43 | 47 |
| | Iteration 7 | | 100 | 10 | 0 | 5 | 22 | 3 | 1 | 0 | 0 | 8 | 100 | 49 | 149 | 67 |
| | Iteration 8 | | | 21 | 3 | 0 | 0 | 0 | 0 | 1 | 4 | 1 | 21 | 9 | 30 | 70 |
| | Iteration 9 | | | | 36 | 3 | 1 | 0 | 1 | 0 | 0 | 3 | 36 | 8 | 44 | 82 |
| | Iteration 10 | | | | | 78 | 2 | 0 | 1 | 0 | 0 | 2 | 78 | 5 | 83 | 94 |
| Program increment 3 | Iteration 11 | | | | | | 60 | 0 | 0 | 0 | 2 | 6 | 60 | 8 | 68 | 88 |
| | Iteration 12 | | | | | | | 57 | 1 | 0 | 0 | 3 | 57 | 4 | 61 | 93 |
| | Iteration 13 | | | | | | | | 54 | 0 | 0 | 5 | 54 | 5 | 59 | 92 |
| | Iteration 14 | | | | | | | | | 63 | 10 | 4 | 63 | 14 | 77 | 82 |
| | Iteration 15 | | | | | | | | | | 52 | 1 | 52 | 1 | 53 | 98 |

the number of open inherited defects that is increasing over time from one iteration to another can be used as a starting point for analyzing the result of the ICE metric with the ultimate goal of detecting in which iteration defects escape.

As best practice, if the software development team does not have the capacity to solve the issues detected during the current iteration, open defects should be planned for implementation in the next iteration. Furthermore, if the number of defects is high, the team can decide to designate one iteration for bug-fixing activities in order to reduce the Defect Debt.

In the long term, improvement actions and measures for continuous process improvement at the iteration level should be defined. Only by improving the PCE for development for the phases used in the iteration can the fault debt and, implicitly, the development costs be reduced. This is how both PCE and ICE can be applied complementarily if the Defect Debt Trend (DDT) indicates that a root cause analysis is necessary.

In the following section, we present the DDT using data from the example mentioned above. This metric shows the cumulative number of open defects in each iteration. It also considers the number of resolved defects from past iterations. Similar to ICE, the DDT metric can also applied be applied at the program level.

DDT iteration $N$ =
(previous iteration defect debt)
+
(total number of defects introduced in the previous iterations and discovered by iteration $N$)
–
(number of defects caused by the previous iterations and solved in iteration $N$) (4)

By analyzing the DDT using equation (4), we can easily identify that in sprint (iteration) 8, the agile team took corrective actions to reduce the number of defects inherited from the previous iterations. We can also forecast that the number of defects should be reduced starting with iteration 11 or in the upcoming iterations.

In , in the rows, we list the defects entered and solved in the same iteration and the number of defects entered in iteration $N$ and discovered in the following iterations $>N$. We apply equation (4) in order to calculate the DDT for each iteration. We consider that the first iteration has a 0 Defect Debt. We present here the data starting with iteration 6. When the report was generated, the last iteration was iteration 15.

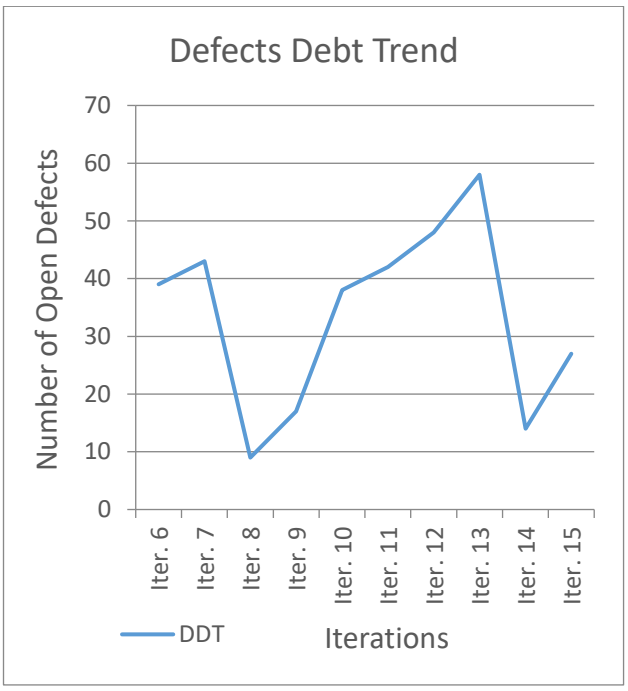


Figure 3. Defect Debt Trend.

## 5. ROOT CAUSE ANALYSIS WITH ICE AND PCE METRICS WHEN MEASURING NUMBER OF CUSTOMER DEFECTS PER LINES OF CODE METRIC

In order to monitor the quality of the delivered product, the Customer Defects per Lines of Code (CDLC) metric can also be used in automotive software development projects by applying Equation (5) [14].

CDLC = number of customer defects/(kilo lines of code) (5)

The number of lines of code can be calculated by using Equation (6):

Source lines of code (SLOC) = total project lines of code – number of empty lines and commented lines (6)

The SLOC is used to determine the size and cost of software development projects. Considering this metric, we can compare the size and complexity of different projects.

Table 2. Defect Debt values for each iteration.

| | | It. 6 | It. 7 | It. 8 | It. 9 | It. 10 | It. 11 | It. 12 | It. 13 | It. 14 | It. 15 | Past iterations defects | Solved defects from past iterations | Defect Debt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Program increment 2 | Iteration 6 | 20 | 3 | 0 | 0 | 1 | 10 | 2 | 0 | 0 | 0 | 4 | 0 | 39 |
| | Iteration 7 | | 100 | 10 | 0 | 5 | 22 | 3 | 1 | 0 | 0 | 4 | 0 | 43 |
| | Iteration 8 | | | 21 | 3 | 0 | 0 | 0 | 0 | 1 | 4 | 16 | 50 | 9 |
| | Iteration 9 | | | | 36 | 3 | 1 | 0 | 1 | 0 | 0 | 8 | 0 | 17 |
| | Iteration 10 | | | | | 78 | 2 | 0 | 1 | 0 | 0 | 21 | 0 | 38 |
| Program increment 3 | Iteration 11 | | | | | | 60 | 0 | 0 | 0 | 2 | 44 | 40 | 42 |
| | Iteration 12 | | | | | | | 57 | 1 | 0 | 0 | 6 | 0 | 48 |
| | Iteration 13 | | | | | | | | 54 | 0 | 0 | 10 | 0 | 58 |
| | Iteration 14 | | | | | | | | | 63 | 10 | 6 | 50 | 14 |
| | Iteration 15 | | | | | | | | | | 52 | 28 | 15 | 27 |

By applying SLOC, the CDLC metric is normalized and can be used for all types of projects, regardless of the complexity and during the entire project lifecycle. As CDLC
values can be compared, they can be used for identifying the lessons learned from completed projects for future projects [15]. When CLDC is measured in agile software development, for every iteration, it can be used for continuous improvement. It provides early feedback of the built-in quality achieved by the delivered working system. It is a highly powerful metric, which allows for the comparison of results between different types of project complexities.

Fault severity classification (major or minor [16]) needs to be defined either at the project or organization level. In order to determine what must be filtered out from the calculation, the business goal(s) and the questions that it is hoped that this metric should answer should be used as the starting point. It should be documented in the metric definition which type of defects will be counted (e.g., all, only major ones, only customer-visible defects).

A company that is using this metric should establish its targets and monitor CDLC in projects accordingly. Companies should take corrective actions [17] and improvement measures to fulfil the defined targets. If deviations appear, a root cause analysis can be undertaken by applying the ICE and PCE metrics. The efficiency of the agile cycles and development phases can be measured using these metrics. In this way, the number of defects discovered by the customer is reduced, and the consequent value and customer satisfaction increase significantly.

In lean agile development [18], the results of the ICE, DDT [19], and CDLC metrics can be used at the retrospective ceremony as process efficiency metrics. By analyzing the output of these metrics, the agile software development team can continuously improve the development and testing processes. Even more so, these metrics can be used to provide objective evidence of improvement.

In order to support organizations and agile software development teams to improve continuously, we detailed in this article the concepts and principles of ICE and DDT metrics [19]. We presented the latest experimental data for ICE, DDT, and how CDLC can be used together with these metrics.

## 6. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

In this article, we showed how classic PCE can be applied to agile software development by considering iterations instead of phases. The ICE metric was defined, and its application was explained in the form of an example. By increasing ICE through continuous process improvement, an agile software development team will not be overwhelmed by the increasing number of defects in the backlog, delivery commitments will be fulfilled, and the quality of the developed product will improve.

We also presented how the ICE metric can be used to analyze the results of the DDT metric, which shows the trend of open defects over time. Even more so, as a result of the improvement measures, the increased rate of ICE should lead to reduced values and lower trends in the DDT. We scaled the ICE metric usage to the PI Level by describing the relevant equation and manner of usage.

Fault debt (remaining open problem reports) from one iteration to another can be monitored and reduced by monitoring, analyzing, and taking appropriate action when measuring the ICE metric. The lowest rates should indicate that a root cause analysis is necessary.

We presented an approach of how to handle defects from previous iterations that are discovered and must be fixed within the current iteration. We also indicated how to track to closure the errors discovered and fixed within the same iteration. If these problem reports are not fixed, the Defect Debt increases, and the development process becomes unsustainable. Variability increases, while for the entire product development system, predictability decreases.

For root cause analyses and to understand an iteration in which defects were not identified in the development or testing phases, the classic PCE for Development (based on the development and test phases used in the iteration) and the PCE for Testing (considering only the development and test phases used in the iteration) can be used.

If organizations want to improve their processes, their products, and customer trust, they should first focus on including in the ICE and DDT metrics only faults that are visible to the end customer, regardless of whether they are critical or not. This suggestion implies that fault severity classification also needs to include customer visibility.

Concerning future research and development, we have clearly proposed how to measure ICE and DDT metrics at the iteration and PI levels. In the future, we intend to investigate how data should be aggregated in order to measure these metrics also at the upper levels of the scaled agile methodologies in the organizations (e.g., the Scaled Agile Framework SAFe® model.

## REFERENCES


[1] Kugler Maag Cie, "Agile in Automotive – State of Practice 2015", May 2015, [Online]. Available: http://www.kuglermaag.com.

[2] VDA QMC Working Group 13/Automotive SIG, "Automotive SPICE Process Assessment/Reference Model, Version 3.0", 2015, pp. 74-75 [Online]. Available: http://www.automotivespice.com/ fileadmin/software-download/Automotive_SPICE_PAM_30.pdf.

[3] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, "Manifesto for Agile Software Development", 2001 [Online]. Available: http://www.agilemanifesto.org.

[4] A. R. Hevner, Phase containment metrics for software quality improvement, Information and Software Technology, 39(13) (1997), pp. 867-877 [Online]. Available: http://www.sciencedirect.com.

[5] D. L. Hallowell, "Six Sigma Software Metrics Maturity, Part 1", Six Sigma Advantage Inc, 2003, [Online]. Available: https://6sigma.com.

[6] I. A. Sandu, A., Salceanu, "Metrics improvement for phase containment effectiveness in automotive software development process", Proc. of the 10th International Symposium on Advanced Topics in Electrical Engineering (ATEE 2017), Mar. 23-25 2017, Bucharest, Romania, pp. 661-666.

[7] Stout Risius Ross, 2016, Apr. 25, Automotive Warranty and Recall Report 2016 [Online], Available: http://www.srr.com.

[8] C. Ebert and R. Dumke: Software Measurement, Springer, Heidelberg, New York, 2007, pp. 245-300.

[9] D. Seidler, T. Southworth, IBM Rational Automotive Engineering Symposium 2013, Source - Herstellerinitiative Software (Audi, BMW, Daimler, Porsche and Volkswagen) 2016, Nov. 21 [Online]. Available: https://www.ibm.com.

[10] Scaled Agile Inc, "SAFe® 4.0" [Online]. Available: http://www.scaledagileframework.com.

[11] Scrum.org, "The Home of Scrum" [Online]. Available: https://www.scrum.org.
[12] J. Shore, S. Warden, The Art of Agile Development, O"Reilly Media Inc, 2008
[13] ScrumCrazy.com, "One way to handle bugs and production support in Scrum" [Online]. Available: http://www.scrumcrazy.com/One+way+to+handle+Bugs+and+Production+Support+in+Scrum.
[14] I. A. Sandu, 2018, "New approach of the customer defects per lines of code metric in automotive SW development applications", XXII World Congress of the International Measurement Confederation (IMEKO) Sept. 3-6, 2018, Belfast, Ireland.
[15] L. W. Walker, "Learning lessons on lessons learned", Paper presented at the PMI® Global Congress 2008 North America, 2008, Denver, Colorado, Newtown Square, Pennsylvania.
[16] K. El Emam, The ROI from Software Quality, Auerbach Publication Taylor & Francis Group, 2005, pp. 14-16.
[17] D. E. Robitaille, J. Rothman, Corrective Action for the Software Industry: A Pragmatic Approach to Effective Problem Solving, 2004, Paton Professional.
[18] D. P. Oosterwal, "The lean machine: How Harley-Davidson drove top-line growth and profitability with revolutionary lean product development", 2012, AMACOM.
[19] I. A. Sandu, A. Salceanu, 2017, "Applications of the Phase Containment Effectiveness metric in automotive industry agile software development", Proceedings of the 22nd IMEKO TC4 International Symposium and the 20th International Workshop on ADC Modelling and Testing Supporting World Development Through Electrical and Electronic Measurements, Sept. 14-15, 2017, Iasi, Romania.