22nd IMEKO TC4 International Symposium & 20th International Workshop on ADC Modelling and Testing
SUPPORTING WORLD DEVELOPMENT THROUGH ELECTRICAL&ELECTRONIC MEASUREMENTS
IASI, ROMANIA, September 14-15, 2017

Applications of the Phase Containment Effectiveness Metric in Automotive Industry Agile SW Development

Ionut-Andrei Sandu[1], AlexandruSalceanu[1]

[1]*Gheorghe Asachi" Technical University of Iasi, Faculty of Electrical Engineering*
siandrei@gmail.com, asalcean@tuiasi.ro

***Abstract* – Phase Containment Effectiveness (PCE) is a very powerful metric which can be used also in the automotive SW development for defects rate minimization. In the classical SW development approach is used the PCE metric in order to detect how efficient is the verification in each of the development phases. In this paper we present the benefits and principles for measureing PCE metric in automotive programs and organizations which adopted Agile SW Development. The acquired advantages are demonstrated by a detailed example of real application on how to measure the PCE metric on Iteration (Sprint) and Program Increment (Scum of Scrums / Scaled Agile) Level.**

***Keywords* – Phase Containment Effectiveness, software quality metric, Agile, Scaled Agile, Iteration, Program Increment**

I. INTRODUCTION

In the automotive industry, the Agile SW Development was adopted by pioneers since 12 years ago, according to a KuglerMaag study published in 2015 [1]. Since then, more and more automotive companies (OEMs and suppliers) which develop software based electronic components are implementing the agile methodology. Main reason is that companies need to keep pace and be flexible with constantly changing requirements, especially in current trend when time to market is decreasing.

Organizations adopting Agile implicitly implement also continuous process improvement, as teams and organizations need to be effective and efficient. Agile process transformation will implicitly trigger improvement actions and measures also for the SW development processes. By this, projects and organizations support and fulfill successfully 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]. This can be translated into deliveries with no faults which affect the end user or faults introduced due to wrong implementation of the requirements.

For defects rate minimization, in the classical SW development approach is used the PCE metric in order to detect how efficient is the verification in each of the development phases. PCE metric answers the following questions:

- how efficient is the verification process?
- which phases escaped defects?
- which phases found/did not find those defects?

But how to answer the above questions for programs and organizations which adopted Agile SW Development?

We here propose a method how to apply the PCE metric in organizations and teams which develop SW for the automotive industry using the Agile SW Development approach.

II. RELATED RESULTS IN THE LITERATURE

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

PCE metric can be also measured in automotive SW Developmentby applying the metric on the specific SW Development and Test Phases.[6].

Faults can be classified into errors and defects, depending on the phase they were injected and the phase they were found in.

Errors are faults discovered in the proper phase they were injected (e.g. design faults caught by design reviews). Defects are faults escaping from their development phase (e.g. design faults caught in code reviews or software test).

Ideally, all faults should be discovered in the phase in which were introduced, leading to an idealistic PCE of 100%. As in the automotive industry the rate of software-

related recalls increased from 5 percent in 2011 to 15 percent in 2015 (Stout Risius Ross study based on data from the United States National Highway Traffic Safety Administration [7]), shows that also in the automotive software development should be attained a better phase containment.

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 [8].

The error correction cost can even increase up to 90 times in post-production phase compared to concept phase [9]. Price of recalls comprises beside fault fixing costs, also legal costs and image costs.

Currently there is no description in the literature how to apply PCE metric in Agile SW Development.

III. DESCRIPTION OF THE METHOD

Agile SW development is executed in iterations (according to Scaled Agile Framework SAFe® model [10] or sprints in SCRUM [11]). At the end of each iteration it should be delivered working SW (ideally faults-free). But if a delivery containing undiscovered faults from Iteration N is used to add on top features for the upcoming deliveries >N, implicitly the undiscovered faults are translated also to these deliveries.

As defects can escape from one iteration to another, the iteration itself can be considered as a phase in the classic PCE metric. Escaped (and basically inherited) faults from one iteration to another can be monitored and reduced by analyzing and taking the proper actions when measuring the PCE for iterations, which can be called Iteration Containment Effectiveness (ICE):

- Iteration Errors: faults caused during iteration N and discovered during Iteration N (e.g.: during Architecture review, code review, SW testing).
- Iteration Defects: faults caused during Iteration N and detected during Iteration > N (next up-coming Iterations) or by the Customer

Total number of faults is obtained after addition (1):

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

Iteration Containment Effectiveness (ICE) can be calculated for each Iteration by applying (2):

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

IV. RESULTS AND DISCUSSIONS

In the following example, we have a Program Increment with four execution iterations. Iteration 11 was the last one executed by the creation date of the report.

By applying (1) and (2), we could calculate the ICE for each Iteration. This is how values from table 2 were obtained for a specific Program.

Table 1. ICE values for a specific Program using Agile

		It.1	It.2	It.3	It.4	It.5	It.6	It.7	It.8	It.9	It. 10	It. 11	Cus-tomer	Total Errors	Total Defects	Total Faults	% ICE
Program Increment 1	Iteration 1	150	40	20	5	0	0	0	0	0	10	0	2	150	77	227	66
	Iteration 2		50	0	10	0	0	0	0	5	2	1	3	50	21	71	70
	Iteration 3			200	50	0	0	0	5	0	0	0	1	200	56	256	78
	Iteration 4				125	10	0	1	0	0	0	3	5	125	19	144	87
Program Increment 2	Iteration 5					10	4	0	1	0	0	5	4	10	14	24	42
	Iteration 6						20	3	0	0	1	10	7	20	41	61	33
	Iteration 7							100	10	0	5	22	8	100	45	145	69
	Iteration 8								21	3	0	0	1	21	4	25	84
Program Increment 3	Iteration 9									36	3	1	3	36	7	43	84
	Iteration 10										78	2	2	78	4	82	95
	Iteration 11											60	8	60	8	68	88

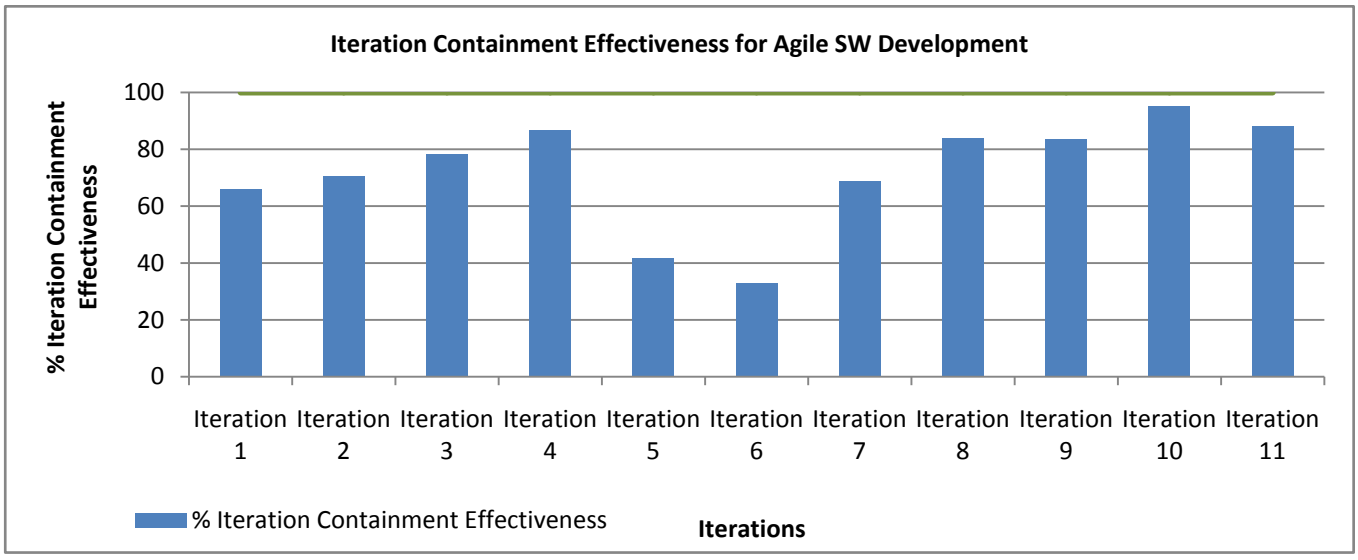

Fig. 1 ICE values for a specific Program using Agile

Also in Agile SW Development can be applied the classical PCE metric on iteration level. For understanding in an iteration which development phases escaped defects and which testing phases missed to detect defects, should be used the classical PCE for Development metric [5] (based on the development and test phases used in the iteration) and classical PCE forTesting metric [5] (considering only the development andtest phases used in the iteration) scaled on iteration level.

As a Program Increment(PI) is consisting of several iterationsand the unit of a Program execution is represented by the PI [10], Iteration Containment Effectiveness can also be translated for Program Increment level by applying the same mechanism. Likethis we obtain the Program Increment Containment Effectiveness (PI CE) which consists of:

- Program Increment Errors: faults discovered during Program Increment N
- Program Increment Defects: faults escaped from Program Increment N and detected during Program Increment > N (next upcoming Program Increments) or by the Customer

$$ProgramIncrementNContainmentEffectiveness = \frac{\text{Program Increment Errors}}{\text{Program Increment Errors+Program Increment Defects}} * 100\% \quad (3)$$

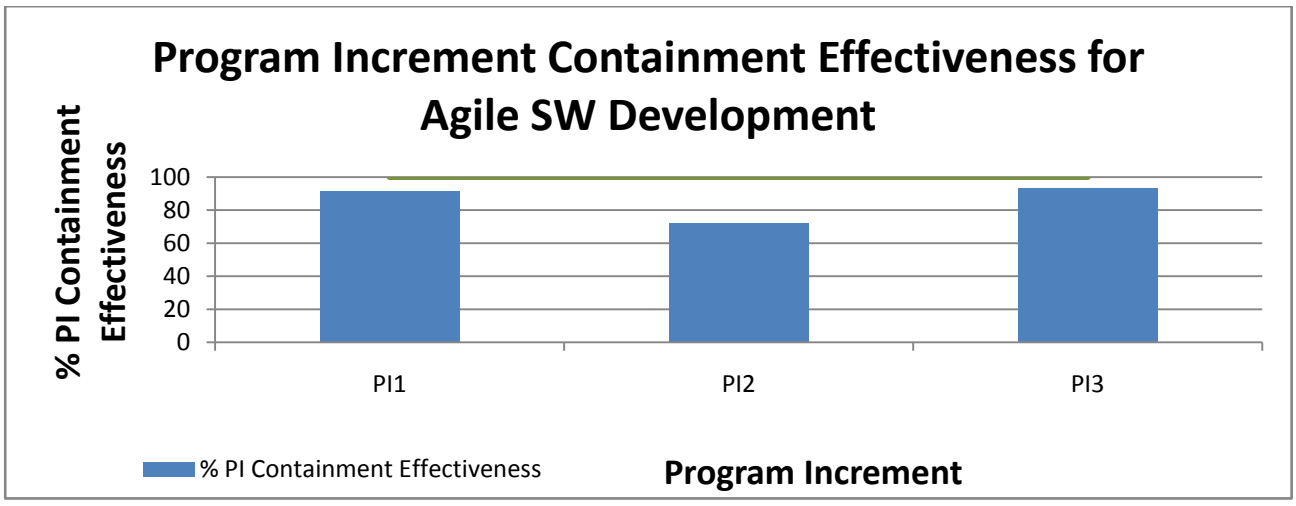

Fig. 2Program Increment Containment Effectiveness values for a specific Program using Agile

ICE and PI CE metrics can be analyzed also together with the trend of open defects for each iteration/each program increment. Basically, if the number of open inherited defects is increasing over time from one iteration to another, it can be used as starting point for the analyses of the result provided by the ICE metric in order to detect which iteration escaped defects. As a best practice, if the team does not have the capacity to solve the detected issues during the current iteration, open defects should be planned to be implemented in the next iteration. Also, if the number of defects is high, the team can decide to plan one iteration dedicated for bug-fixing activities in order to reduce the defect debt. On long term, improvement actions and measures for continuous process improvement on iteration level should be defined. Only by improving the PCE for development for the phases used in the iteration, we can reduce the faults debt and implicitly the development costs. This is how both PCE and ICE can be applied complementary if DDT indicates that root cause analyses is necessary.

In the following section, we present the Defects Debt Trend (DDT) using the data from the example mentioned above.This metric shows the cumulated number of open defects in each iteration. It also considers the number of solved defects from the past iterations.Similarly, to the ICE, DDT metric can be also applied on the Program Level.

DDT Iteration N =
(Previous Iteration Defects 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)

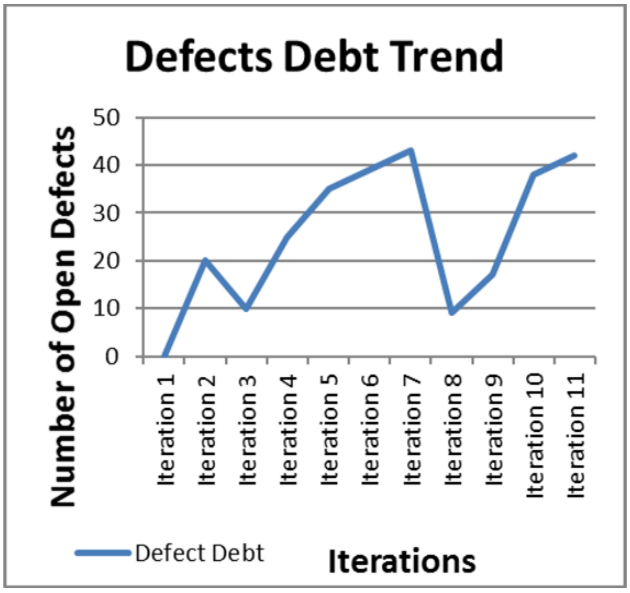

Figure3. Defect Debt Trend

By analyzing the Defect Debt Trend, we can easily identify that in sprint (iteration) 8 the agile team took corrective actions in order to reduce the number of defects inherited from the previous iterations. We can also make a forecast that the number of defects should be reduced starting with iteration 11 or with the next upcoming iterations.

In the table below on the rows welisted the defects enteredand solved in the same iteration and the number of defects entered in iteration N and discovered in following iterations >N. We applied formula (4) in order to calculate DDT for each iteration. We considered that the first iteration had 0 defect debt. When the report was generated the last iteration was It. 11.

Table 2. Defect Debt Values for each Iteration

		It.1	It.2	It.3	It.4	It.5	It.6	It.7	It.8	It.9	It. 10	It. 11	Past Iterations Defects	Solved Defects From PastIterations	Defect Debt
Program Increment 1	Iteration 1	150	40	20	5	0	0	0	0	0	10	0	0	0	0
	Iteration 2		50	0	10	0	0	0	0	5	2	1	40	20	20
	Iteration 3			200	50	0	0	0	5	0	0	0	20	30	10
	Iteration 4				125	10	0	1	0	0	0	3	65	50	25
Program Increment 2	Iteration 5					10	4	0	1	0	0	5	10	0	35
	Iteration 6						20	3	0	0	1	10	4	0	39
	Iteration 7							100	10	0	5	22	4	0	43
	Iteration 8								21	3	0	0	16	50	9
Program Increment 3	Iteration 9									36	3	1	8	0	17
	Iteration 10										78	2	21	0	38
	Iteration 11											60	44	40	42

V. CONCLUSIONS

We showed how classical PCE can be applied in Agile by considering iterations instead of phases. ICE metric definition was defined and its application was explained by the presented example. By increasing ICE through continuous process improvement, the agile team will not be overwhelmed by the increasing number of defects in the backlog, delivery commitments will be fulfilled and quality of the developed product will increase.

We also present how ICE metric can be used in order to analyze the result of the Defects Debt Trend metric which shows the trend of open defects over time. More than this, as a result of the improvement measures, an increased rate of ICE should lead to reduced values and lower trend in Defects Debt Trend.We scaled the ICE metric usage also on the Program Increment Level by describing the formula and way of usage.

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

For root cause analyses and for understanding in an iteration which development phases escaped defects and which testing phases missed to detect defects, can be used the classical PCE for Development (based on the development and test phases used in the iteration) and PCE for Testing (considering only the development and test phases used in the iteration).

If organizations want to improve their processes, their products and especially the customer trust, they should focus in a first step in counting for the ICE and DDT metric only the faults which are visible to the end customer, independent if these are critical or not. This implies also that fault severity classification needs to include also the customer visibility.

Further development: We presented how to measure ICE and DDT metric on iteration and program increment levels. It can be also investigated how to measure these metrics on the upper levels of the Scaled Agile.

REFERENCES

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

[2] VDA QMC Working Group 13 / Automotive SIG. (2015, July 16).*Automotive SPICE Process Assessment / Reference Model, Version 3.0,* pp. 74-75,[Online]. Available:http://www.automotivespice.com/fileadmin/softwaredownload/Automotive_SPICE_PAM_30.pdf

[3] Beck, K., Beedle, M., Bennekum, 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].

[4] A. R. Hevner. (1997). Phase Containment Metrics for Software Quality Improvement. *Information and Software Technology* [Online]. 39(13), pp. 867–877. Available: http://www.sciencedirect.com.

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

[6] Sandu, I.-A., Salceanu, A., 2017, *Metrics Improvement for Phase Containment Effectiveness in Automotive Software Development Process*, Proceedings of the10-th International Symposium on ADVANCED TOPICS IN ELECTRICAL ENGINEERING (ATEE 2017), 23-25 March, Bucharest, Romania, pp.661-666.

[7] Stout Risius Ross. (2016, April 25).Automotive Warranty & 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. (2016, November 21).IBM Rational Automotive Engineering Symposium 2013, Source - Herstellerinitiative Software (Audi, BMW, Daimler, Porsche and Volkswagen). [Online]. Available: https://www.ibm.com.

[10] Scaled Agile, INC. *SAFe® 4.0* [Online]. Available: http://www.scaledagileframework.com/.[Accessed 03 Apr. 2017]

[11] Scrum.org.*The Home of Scrum* [Online]. Available: https://www.scrum.org/. [Accessed 03 Apr. 2017]