IMPLEMENTATION OF PHASE CONTAINMENT EFFECTIVENESS METRIC USING DEFECT-INJECTION MODEL AND CAUSAL ANALYSIS OF DEFECTS

Ms. Shweta Sharma

PhD. Research Scholar, Department of Computer Science, Mewar University Chittorgarh, Rajasthan, India
bhardwaj.shweta28@gmail.com

Dr. S. Srinivasan

Head of Department, Department of Computer Science and Applications, PDM University, Bahadurgarh, Haryana, India
dss_dce@yahoo.com

Abstract

The most significant objective of software engineering is to improve productivity of software by developing quality software products diminishing the software cost and time utilizing the best engineering and management practices. The quality of software is inversely proportional to the number of defects. The defects have the probability of multiplying and replicating themselves to the next phases of the development life cycle if not detected and removed in initial phases. Phase Containment Effectiveness metric is used to detect defects at current phase and also the escaped defects in the previous phases. This paper has used a defect injection model to implement the phase containment effectiveness metric. The implementation has been done on the real software development project. The results of the implementation have been shown using MS-excel charts and tables. The paper also provides a causal analysis of defects so as to diagnose the reasons behind the occurrence of defects.

Keywords: - **Defect Phase Containment, Cumulative Phase Containment, Injection Model, NGT, Fishbone Diagram**

1. Introduction

A defect is usually the manifestation of bugs and errors. It can also be because of failure of a component in a computer program or entire software [1]. The flaws or faults in a component can never let the software perform its intended function [2]. For example if an operating system is full of bugs and defects then it will never let any application software run smoothly on it [3]. The defects in the software arise when clients' expectations are not met. These defects need to be predicted and removed using software metrics to maintain software quality [4] [5]. To improve software quality, the project managers need defect prediction metrics and models so that valuable resources can be utilized efficiently [6] [7]. Predicting defects is a challenging job for the project manager and his team mates [8]. Generally, the quality of a software work product is determined at the end of the testing phase based on the total number of defects detected throughout the project life cycle. The actual number of detected defects is compared against a targeted one and a decision is taken regarding the suitability of the product to be delivered [9]. This targeted figure is an indicator of the expected number of product defects. The defect injection model uses the method of injecting defects at basic phases of lifecycle and then detection method is used to check and verify that the previously injected defects have been detected or escaped. The factors influencing the DI (Defect Injection) for example requirements, domain knowledge, product complexity, process maturity and documentation quality must be taken into account while injecting the defects; similarly DD (Defect Detection) factors such as test planning, change control, test environment and team distribution must also be considered [10]. This clearly evaluates the efficiency of the phase containment metric by comparing the predicted defects with the actual defects. Once the difference is known, the trend analysis of the past few years can be drawn to check the effectiveness of the metric. Finally, the causes of defects can be determined by using techniques like Pareto chart, NGT and fishbone or ishikawa analysis.

2. Related Literature

The motive of this research article is concerned with the development of a decision-based model to assist project managers in controlling cost, schedule and software quality. The existing literature studies also reinforce that the bugs and errors should be detected and corrected at early phases of software life cycle [11]. PCE (Phase Containment Effectiveness) metric gives an early warning signal to dedicated and competent project managers. It allows them time to take an immediate action such as by adding resources to the system testing activities. The most reputed existing model for defect removal and quality improvement is COQUALMO (Constructive Quality Model) which is classified into defect introduction and defect removal sub-models [12]. This model works in integration with COCOMO (Constructive Cost Model) to determine the cost spent in defect removal using the quality assurance activities such as "Automated Analysis, Review and Testing" [13]. An existing research study has presented the use of PCE metric on different development phases to detect 139 and 80 defects on two software versions hypothetically called as Reliant Release 4.0 and Release 3.0 respectively [14]. This use of phase containment metrics model is formally defined and demonstrated by Hevner (1997) to improve software quality and process [15]. There is another study of the Phase Containment Effectiveness method in two different applications in the automotive software development process: one for development phases and the other for test phases [16]. The existing literature also shows that PCE metric can be well applicable for agile development where the development takes place in the form of increments or iterations [17]. For our research purpose, the basic phases of software development life cycle have been divided into: "Analysis, Coding, Design, Testing (Unit Testing, System Testing and Acceptance Testing): and the defects are injected at all these phases using Defect Injection Model. Apart from applying the phase effectiveness Metric, the Cumulative Phase Effectiveness metric have also been implemented to check and detect the escaped defects in all previous phases. The paper also presents the causal analysis techniques to determine the root causes of defects.

3. Data Collection

Although PCE for the delivered product is known only at the end of the acceptance, collecting this metric at the end of every phase enables an early understanding of escaped defects from earlier phases. This can be used to initiate appropriate defect prevention actions in the ongoing phases. For example, at the end of Design phase, we will calculate PCE metrics for Analysis phase; at the end of Coding phase, we will calculate PCE metrics for Analysis and Design phases. In cases where modules are delivered and acceptance testing is conducted at module level, we can compute this metric at module level provided the modules are not too interdependent. However, these metrics will be finalized after acceptance. PCE need not be collected if the project starts from the Construction phase only (where the client is responsible for the Analysis and Design phase).

4. Using the Metric

From the data collected, PCEi for a particular phase may be calculated as follows:

$$\text{PCEi} = \frac{\text{No. of phase i defects found during phase i review/test}}{\text{No. of defects with source i found in all the phases}} * 100\ \%$$

Fig. 1. Phase Containment Effectiveness

While PCE_i concentrates on the efficiency of detecting phase i defects in phase i reviews, cumulative PCE_i concentrates on the efficiency of the phase i review in detecting escaped defects from all earlier phases in addition to those injected in phase i. This is used to monitor the percentage of defects with origin attributed to a particular phase but detected in subsequent project phases. Ideally, all defects of a phase should be detected during the review process of that phase, i.e. PCEi should be 100%.

For example, the number of defects found in the analysis phase, which can be attributed to analysis activity is 10. Number of defects found in the design phase, which can be attributed to analysis activity is 5. Number of defects found in the construction phase, which can be attributed to analysis activity is 5. Then at the end of the construction phase, the phase containment for the analysis phase is 10/(10+5+5) * 100= 50%. During acceptance, 5 more analysis defects may be found which makes the PCE for the analysis phase as (10)/(20+5) * 100 = 40%.

5. Implementation and Results Discussion

The Defects are injected during development phase as shown in the figure below. Now the defect detection procedure will check for the presence of defects. The defects which get detected during detection process are repaired and removed. But some defects which are not detected get escaped and need to be detected in next phases in the cumulative form (current phase's defects and defects of previous phases).

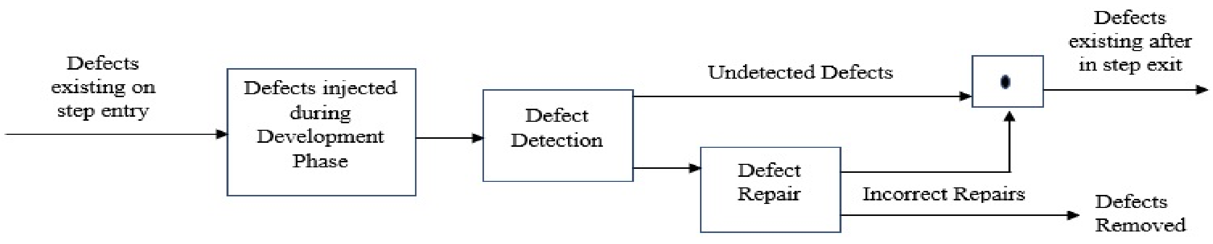
Defects existing on step entry
Defects injected during Development Phase
Defect Detection
Undetected Defects
Defect Repair
Incorrect Repairs
Defects existing after in step exit
Defects Removed

Fig. 2. Defect Injection and Detection Model

5.1 Injection Phase: **-** The objective of this metric is to increase defects so as to determine the effectiveness of the defect detection process, hence the defects are introduced in the injection phase. Injection phases will remain the usual SDLC phases such as: -

a. Analysis
b. Design
c. Coding
d. Unit Testing (UT)
e. System Testing (ST)
f. Acceptance Testing (AT)

5.2 Detection Phase: **-** The following items will be used for calculation defect detection percentage.

i. Total Defect Containment: - This will be the count of total defects contained in each phase.
ii. Phase Containment Effectiveness: - This will be the percentage of the defects detected in a particular phase in respect to the total defects injected in each phase.

The following table contains the data collected related to this metric: -

Injection Phase	**Detection Phase ⟶**							
	Analysis	**Design**	**Coding**	**UT (Unit Testing)**	**ST (System Testing)**	**AT (Acceptance Testing)**	**Total errors injected in each phase**	**Phase Containment Effectiveness**
Analysis	64	12	6	0	2	0	84	**76.19%**
Design	NA	43	12	3	1	0	59	**72.88%**
Coding	NA	NA	84	11	7	5	107	**78.50%**
UT	NA	NA	NA	7	0	3	10	**70.00%**
ST	NA	NA	NA	NA	7	0	7	**100.00%**
AT	NA	NA	NA	NA	NA	0	0	**100.00%**
Total Errors detected in all phases	64	55	102	21	17	8	267	

Table 1. Defect Data and PCE$_i$ for A Real Time Project

The above table shows the defects injected and detected at each phase of the development life cycle. Then the phase containment effectiveness percentage of each phase can be computed by dividing the detected defects of each phase with the total errors injected in each phase and multiplying by 100. For example the Phase Containment Effectiveness of Analysis is calculated as 76.19% ((64 / 84)*100 = 76.19%). Similarly, the Phase Containment Effectiveness of Design phase is 72.88 ((43 / 59) *100 =72.88%).

Once the Phase Containment Effectiveness is computed for all phases, the results can be analyzed by drawing the defect injection curve as shown below in the figure.

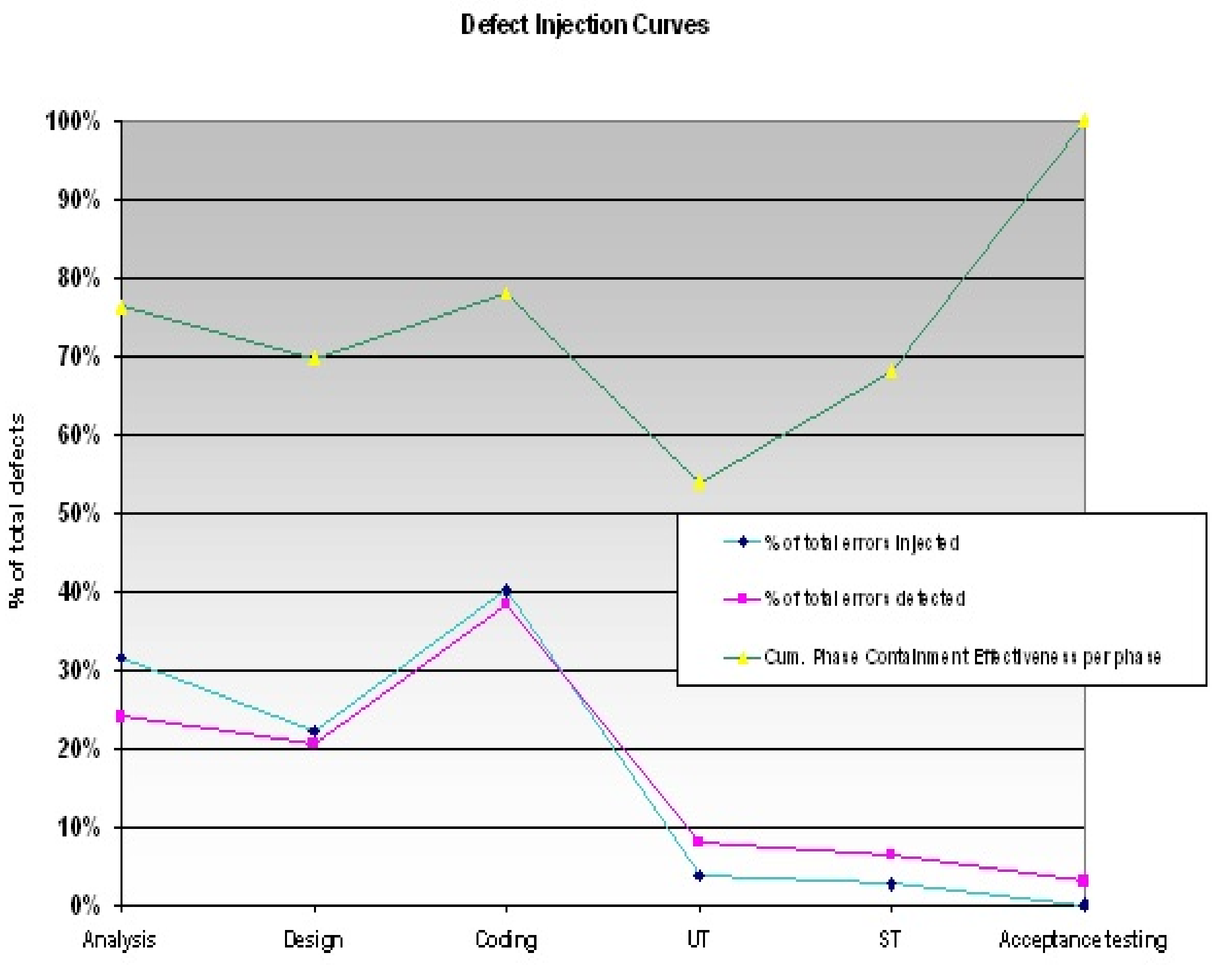

Fig. 3. Defect Injection and Detection Curves

As per the above figure the curve for Phase Containment Effectiveness meets at 100% during AT (Acceptance Testing) implying that all escaped defects from earlier phases have been detected now.

The next section shows the computation of Cumulative Phase Containment Effectiveness at each phase.

Phase	**Errors injected in the phase**	**Errors detected in the phase**	**Cumulative Injection up to the phase**	**Cumulative Detection up to the phase**	**Net Escapes up to and including the phase**	**% Of total errors injected**	**% Of total errors detected**	**Cumulative Phase Containment Effectiveness per Phase**
Analysis	84	64	84	64	20	31.46%	23.97%	**76.19%**
Design	59	55	143	119	24	22.10%	20.60%	**69.62%**
Coding	107	102	250	221	29	40.07%	38.20%	**77.86%**
UT	10	21	260	242	18	3.75%	7.87%	**53.85%**
ST	7	17	267	259	8	2.62%	6.37%	**68.00%**
AT	0	8	267	267	0	0.00%	3.00%	**100%**
Total	267	267	267	267	99	100.00%	100.00%	

Table 2. Cumulative Phase Containment Effectiveness computation

In above figure, the numbers of defects injected in analysis phase are 84 and total numbers of defects up to all phases are 267. Hence % of total errors injected is 31.46 ((84 / 267) *100 = 31.46%). Similarly, the percentage of errors detected in analysis phase is 23.97% ((64 / 267)*100).

$$\text{CPCEi} = \frac{\text{Number of defects found in Phase i review}}{\text{Number of defects detected in phase i + Cumulative Number of defects escaped from all previous phases}}$$

Fig. 4. Cumulative PCE calculation

$$\text{CPCE}_i\,(\text{Analysis}) = (64/\,(64+20)) * 100 = 76.19\%$$

Similarly, for design phase, the Cumulative Phase Containment Effectiveness percentage will be calculated as 69.62% ((55 / (55+24))*100). Once the phase containment and cumulative phase containment metrics are used to capture defects; the defects can be compared with the lower and upper tolerance limits of defects which are always pre-fixed by the organization. A demonstrative example is shown in figure below-

#	PHASE	PREDICTED	LOWER LIMIT	UPPER LIMIT	ACTUAL
1	Analysis	27	18	35	30
2	Design	62	43	81	72
3	Construction & Unit Testing	124	82	165	115
4	System Testing	38	20	56	35
5	Acceptance Testing	12	7	17	4

Table 3. Actual and Predicted Defect Comparison on sample data

A trend analysis over the periods of years can be drawn which helps in overall quality improvement. High defect density and low defect density deliverables need to be monitored [18]. There can be correlation between the development practices of an organization and defect density. This correlation can then be used for the improvement of software development practices for good return on investments [19]. The trend analysis on the basis of the above table is shown in the figure below.

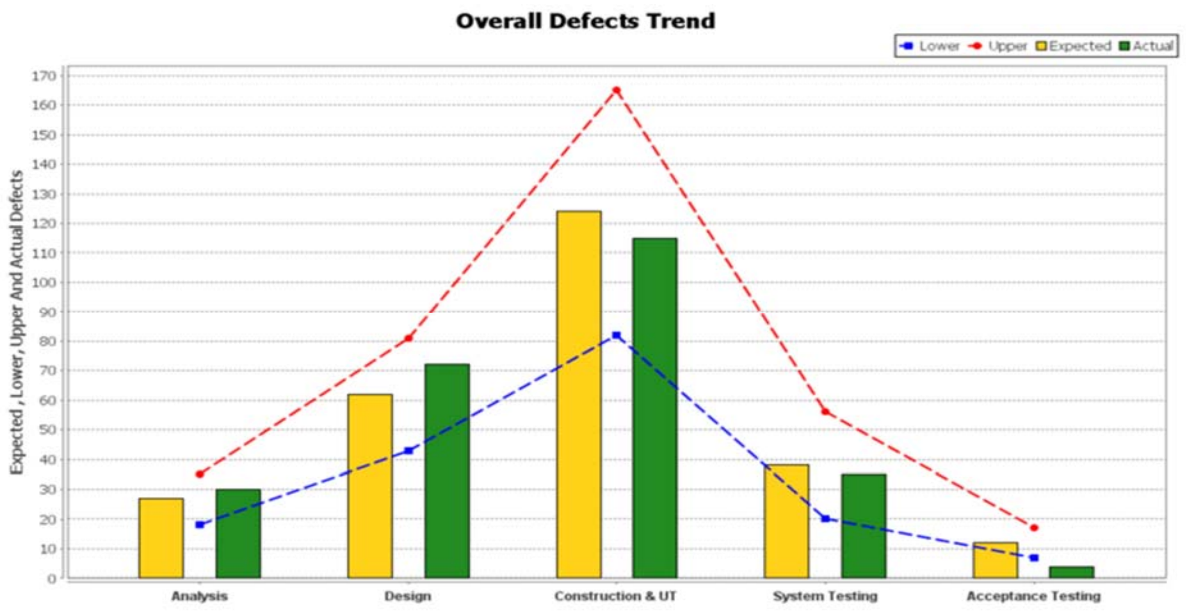

Fig. 5. Overall Defect Trend Analysis

6. Causal Analysis of Defects

Causal analysis is a method to find the root causes of a defect or problem. Causal analysis helps to find the fundamental reasons as to why defects have occurred. Once the root causes are known, corrective and preventive actions can be taken to avoid the recurrence of the same or possibly other defects [20].

6.1 Pareto Analysis: - The Pareto chart is used to segregate the insignificant causes from the significant ones [21]. The intent is to:

- Focus attention on critical factors that are most significant to a problem
- Display which path or problems to pursue
- To find where to concentrate for biggest improvements

Constructing the Pareto Chart: - The Pareto analysis is based on the "80/20 rule". This implies that the reasons for 80% of the defects are 20% of the causes and hence those causes must be focused on to obtain the maximum benefits [22]. The six steps in constructing a Pareto Chart are:

1. List the activities or causes in a table and count the number of times each occurs
2. Place these in descending order of magnitude in the table
3. Calculate the total for the whole list
4. Calculate the percentage of the total that each cause represents
5. Draw a Pareto chart with the vertical axis demonstrating the percentage and the horizontal axis presenting the causes. If cumulative percentage of all the causes is required then the curve for cumulative causes can also be drawn.
6. Interpret the results.

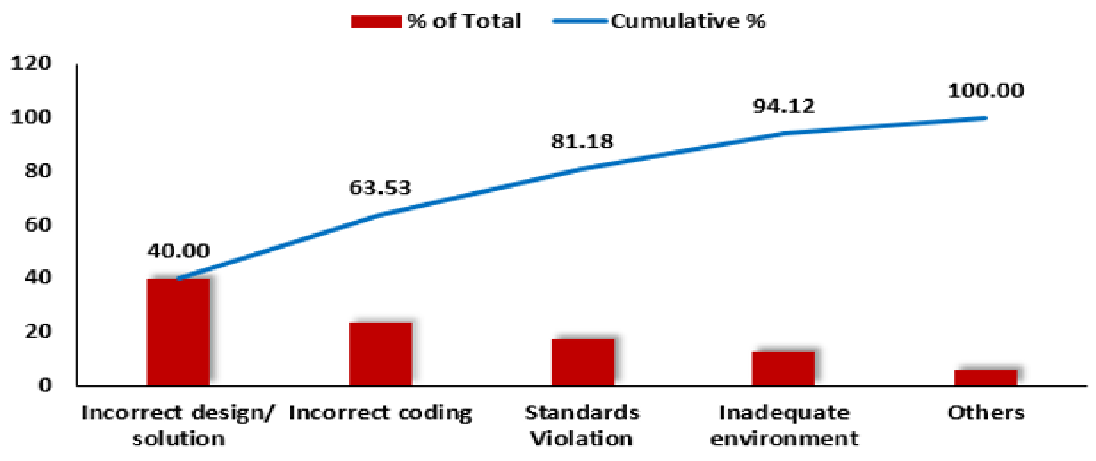

Fig. 6. A Pareto chart drawn with sample data to show the root causes of defects

6.2 Nominal Group Technique (NGT): - Nominal Group Technique (NGT) is used to accumulate opinions from a group of people and recognize the level of support within the group for those opinions [23]. The NGT can be used:

- when a consensus is required to determine a conclusion on a collective course of action
- when one person is dictating a group and when the others need to be involved
- at the end of a brainstorming session to bring together and "harvest" opinions

***6.3 NGT typically includes the following simple steps** [24]:*

1. Decide and select a group leader to record the opinions and ideas (say, on a flip chart).
2. Each participant is requested to provide an idea. At this stage, the ideas are not evaluated, judged or criticized. The ideas are made visible for all in the group.
3. Step 2 is repeated until the group has been exhausted of ideas or a specified time limit is reached.
4. Group members are allowed to ask questions and seek clarifications on the ideas.
5. Every participant is asked to write five ideas and assign rank to the ideas from 1 to 5, assigning 1 to the idea which is of least importance and 5 to the most significant one. Finally, the participants are required to record their scores on the given flip charts next to the idea being judged.
6. It is the responsibility of the group leader to sum up all the scores and prepare the results.

Then the participants of the group leave the session dedicated to the conclusion. The technique might be varied as follows:

- Group members can be given colored markers to do the final scoring
- Group members can be given 5 votes and they can allocate these between ideas

Who	Idea 1	Idea 2	Idea 3	Idea 4	Idea 5
Person 1	1	3	4	2	0
Person 2	3	1	0	0	2
Person 3	0	0	1	3	0
Person 4	1	0	0	0	4
Person 5	0	2	0	0	3
Person 6	3	0	1	2	4
Person 7	0	0	2	0	1
Person 8	3	1	0	0	0
Person 9	0	0	4	2	0
Person 10	1	0	4	0	3
Total	12	7	16	9	17

Table 4. An example of NGT technique performed on sample data

On observing data from the above table, "Idea 5, Idea 3, Idea 1, Idea 4 and Idea 2" will be the order for concentration.

6.3 Fishbone (Cause-Effect or Ishikawa) Analysis: - The Fishbone Diagram is also known as the Ishikawa Diagram or Cause-Effect Diagram. This diagram is used to evaluate the impacts to detect the probable causes of those impacts and to indicate possible areas for gathering the data [25]. This approach can be used by the project team to find possible solutions to a problem. The intent is:

- To analyze the root causes of the problem: effect is the quality problem under investigation.
- To prevent "tunnel vision" by expanding beyond potential causes.
- To help in determining what to collect, what to focus on and to verify the root causes.

The following steps can be carried out in constructing a Fishbone Diagram:

- Brainstorm all probable causes of the problem or effect chosen for analysis. Use NGT to arrive at group consensus.
- Write down the problem statement in a box at the right-hand side of the diagram.
- Identify the categories in which you expect to find causes of the problem. Label the major bones of the fish with these categories.

Typical major categories are: - Machines, Methods, Materials, People, Information, Environment, Measurements

- The causes are supposed to be written on the diagram under the classifications chosen as "smaller bones"

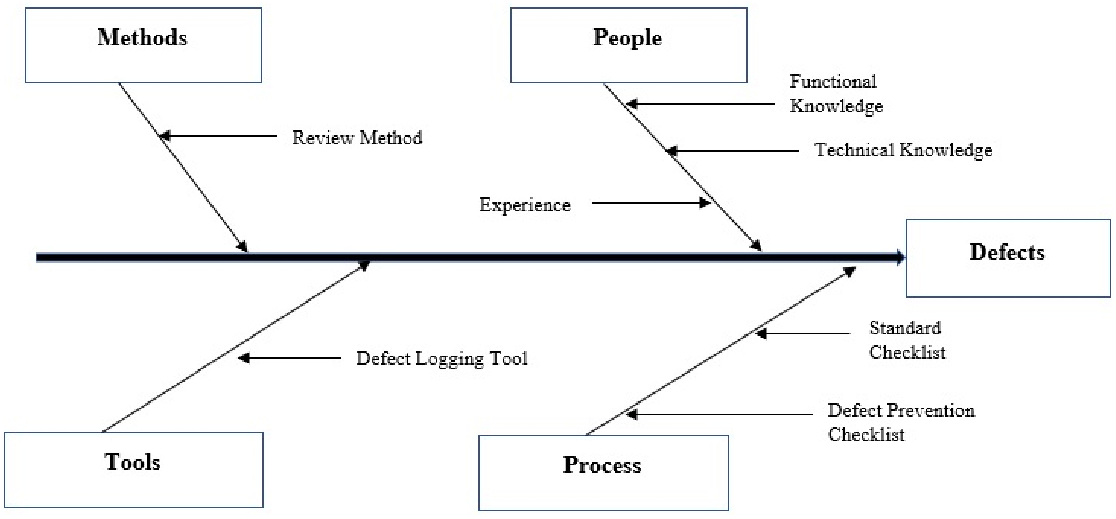

Fig. 7. A Fishbone diagram drawn to know the root causes of Defects

The above diagram states that the causes of the defects related to methods, people, tools and machinery and the process followed. The root causes can also be inadequate training and skills, breakdown of communication, mistakes in manual procedures, process deficiency, inadequate resource allocation, and lack of clarity on product requirements, incomplete, ambiguous or unclear contractual requirements and so on [26]. Hence the causal analysis is highly required to identify the causes of the defects so as to prevent the defects from propagating on all phases of development.

7. Conclusion

In this article, we have implemented the defect containment effectiveness metric on a real time software development project data. The defects which are escaped at initial phases can carry forward themselves in next phases. The paper also reinforces that the cumulative PCE can be effectively applied in cases which measure the efficiency of a phase in detecting escaped defects from all previous phases in supplement to the defects injected in the current phase. The defect injection model is used for the above purpose to inject defects at development phases. Conclusively, the defect detection will be considered as successful if the percentage of Cumulative Phase Containment Effectiveness is either 100% or closer to 100%. The article also suggested the techniques which can be used to determine the reasons for defect occurrence. As far as future implications are considered, the implementations of Phase Containment Effectiveness and Cumulative Phase Containment Metrics discussed in this paper can also be tried for other approaches such as Object Oriented and Aspect Oriented Software Development.

References

[1] W. S. Humphrey, “Bugs or defects?” CMU SEI, vol. 1, 1999.
[2] P. K. Singh, D. Agarwal, and A. Gupta, “A systematic review on software defect prediction,” in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2015, pp. 1793–1797.
[3] M. Sullivan and R. Chillarege, “Software defects and their impact on system availability: A study of field failures in operating systems,” in *FTCS*, 1991, vol. 21, pp. 2–9.
[4] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, “Choosing software metrics for defect prediction: an investigation on feature selection techniques,” *Softw. Pract. Exp.*, vol. 41, no. 5, pp. 579–606, 2011.
[5] M. S. Rawat and S. K. Dubey, “Software defect prediction models for quality improvement: a literature study,” *Int. J. Comput. Sci. Issues IJCSI*, vol. 9, no. 5, p. 288, 2012.
[6] Y. Kamei and E. Shihab, “Defect prediction: Accomplishments and future challenges,” in *2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER)*, 2016, vol. 5, pp. 33–45.
[7] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, “An empirical study on software defect prediction with a simplified metric set,” *Inf. Softw. Technol.*, vol. 59, pp. 170–190, 2015.
[8] E. A. Felix and S. P. Lee, “Integrated approach to software defect prediction,” *IEEE Access*, vol. 5, pp. 21524–21547, 2017.
[9] D. Bowes, T. Hall, and J. Petrić, “Software defect prediction: do different classifiers find the same defects?,” *Softw. Qual. J.*, vol. 26, no. 2, pp. 525–552, 2018.
[10] J. Jacobs, J. Van Moll, R. Kusters, J. Trienekens, and A. Brombacher, “Identification of factors that influence defect injection and detection in development of software intensive products,” *Inf. Softw. Technol.*, vol. 49, no. 7, pp. 774–789, 2007.
[11] M. Sherriff, N. Nagappan, L. Williams, and M. Vouk, “Early estimation of defect density using an in-process Haskell metrics model,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–6, 2005.
[12] S. Chulani and B. Boehm, “Modeling software defect introduction and removal: COQUALMO (COnstructive QUALity MOdel),” Citeseer, 1999.
[13] L. LAZIĆ, I. \DJOKIĆ, and S. MILINKOVIĆ, “Estimating cost and defect removal effectiveness in SDLC testing activities,” *Proc. INFOTEH-JAHORINA*, vol. 12, pp. 572–577, 2013.
[14] R. Seider, “Implementing phase containment effectiveness metrics at Motorola,” *Crosstalk*, vol. 19, no. 11, pp. 12–14, 2006.
[15] A. R. Hevner, “Phase containment metrics for software quality improvement,” *Inf. Softw. Technol.*, vol. 39, no. 13, pp. 867–877, 1997.
[16] I.-A. Sandu and A. Salceanu, “Metrics improvement for Phase Containment Effectiveness in automotive software development process,” in *2017 10th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, 2017, pp. 661–666.
[17] I. A. Sandu and A. Salceanu, “New Approach on the Agile Cycles Containment Effectiveness Metrics in Automotive SW Development,” *ACTA IMEKO*, vol. 7, no. 4, pp. 3–8, 2019.
[18] S. M. A. Shah, M. Morisio, and M. Torchiano, “An overview of software defect density: A scoping study,” in *2012 19th Asia-Pacific Software Engineering Conference*, 2012, vol. 1, pp. 406–415.
[19] S. M. A. Shah, M. Morisio, and M. Torchiano, “An overview of software defect density: A scoping study,” in *2012 19th Asia-Pacific Software Engineering Conference*, 2012, vol. 1, pp. 406–415.
[20] B. Johnson, Y. Brun, and A. Meliou, “Causal testing: understanding defects’ root causes,” in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 87–99.
[21] D. Radson and A. H. Boyd, “The pareto principle and rate analysis,” *Qual. Eng.*, vol. 10, no. 2, pp. 223–229, 1997.
[22] C. Brooks, “What Is a Pareto Analysis?,” *Bus. News Dly. Sr.*, vol. 29, 2014.
[23] A. Dobbie, M. Rhodes, J. W. Tysinger, and J. Freeman, “Using a modified nominal group technique as a curriculum evaluation tool,” *Fam. Med.-Kans. CITY-*, vol. 36, pp. 402–406, 2004.
[24] R. B. Dunham, “Nominal group technique: a users’ guide,” *Madison Wis. Sch. Bus.*, vol. 2, 1998.
[25] S.-S. Li and L.-C. Lee, “Using fishbone analysis to improve the quality of proposals for science and technology programs,” *Res. Eval.*, vol. 20, no. 4, pp. 275–282, 2011.
[26] W. Al-Ahmad, K. Al-Fagih, K. Khanfar, K. Alsamara, S. Abuleil, and H. Abu-Salem, “A taxonomy of an IT project failure: Root causes,” *Int. Manag. Rev.*, vol. 5, no. 1, p. 93, 2009.

Author's Profile

Shweta Sharma: She is working as an Assistant Professor in Computer Science Department in Government College for Girls, Sector-14, Gurugram. She has 13 years of teaching experience. She is pursuing Ph.D. (Computer Science) from Mewar University, Rajasthan. She did BCA from University College Kurukshetra (KUK) and MCA from Department of Computer Science and Applications from Maharshi Dayanand University, Rohtak. She has also done MPhil (Computer Science) from Ch. Devilal University, Sirsa. She has also done B.Ed in computer science and English from Maharshi Dayanand University. Her areas of research are Software Engineering, Data Base Management System, Computer Networks and Cybersecurity.

Prof. (Dr) S Srinivasan: He is currently working as a Head of the Department of Computer Science and Application in PDM University, Bahadurgarh. He has a vast experience of nearly 40 years including 22 years in teaching and 18 years in Industry. He presented Research papers in various National and International conferences. He was a Research Advisor of Bharathidasan University, Trichy, Tamilnadu. He has produced twelve PhDs and guiding six research students. He published 42 papers in a National and International Journals. His current research field is the area of Artificial Intelligent especially in Multi-Agent System Technology and its applications.