A Framework for Software Defect Management Process in Software Quality Assurance

Mehreen Sirshar
Head Of Department
Software Engineering Department
Fatima Jinnah Women University
Rawalpindi,Pakistan
msirshar@gmail.com

Jumana Noor
Software Engineering Department
Fatima Jinnah Women University
Rawalpindi,Pakistan
jumanakhan98@gmail.com

Raima Rashid
Software Engineering Department
Fatima Jinnah Women University
Rawalpindi,Pakistan
raimarashid610@gmail.com

Muneeba Daud
Software Engineering Department
Fatima Jinnah Women University
Rawalpindi,Pakistan
muneebadaud97@gmail.com

***Abstract*—The success of a software system is highly dependent upon its quality which is a very critical aspect and is the ultimate goal to be achieved. Defect Management is the most important process in ensuring software quality, it involves defect detection, defect categorization, defect prioritization, defect removal and defect verification. The main purpose of this process is to develop and produce defect free or least defected software systems of high quality. It plays a vital role in gaining customer confidence and satisfaction which is essential for the reputation of the organization developing the system. The central objective of this study is to explain the importance of defect management and its impact on the quality of the software along with various defect management techniques to support the objective.**

Keywords—defect management, defect detection, defect categorization, defect prioritization, defect removal defect verification

I. Introduction

The quality of a software system can be termed as the measure of number of defects in it. A software defect is a mistake or flaw in the system that prevents it from behaving in the intended or programmed way, it may be caused due to an incorrect step, process, or data definition in a computer program, misinterpreted or mistaken requirement or any other fault in the development which leads the system to give undesired or unexpected output which is other than the actual desired output. Such systems are known as defected software systems and are needed to be fixed by the testing or quality assurance team.

In order to produce high quality software systems with least number of defects, effective defect management strategies must be applied to the software development cycle. The development of a software does not only require programming or coding rather handling and managing defects is of utmost importance while developing it. Thus, formal defect management process must be enforced to efficiently handle defects. The defect management process (DMP) proposed in this study involves five levels; Defect Discovery, Defect Analysis, Defect prioritization, Defect removal and Defect verification. This framework is used to handle the defects identified or present in the system.

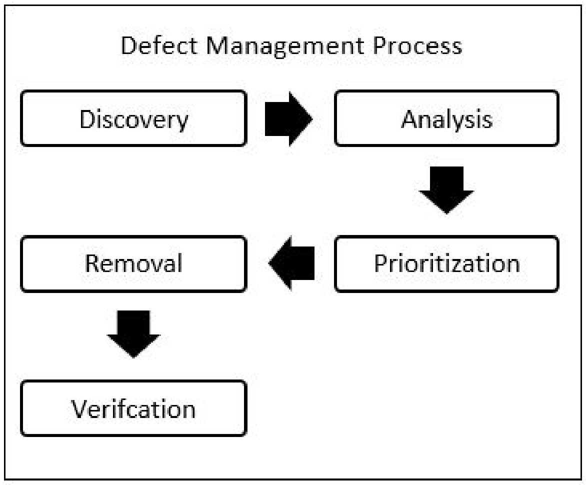

Fig. 1. Block Diagram of Defect Management Process

The Block diagram of DMP shows the sequence steps involved in managing and resolving defects. The first step is to discover the defect, that whether there is a defect present in the system or not. Various techniques can be used to discover a defect for example alpha or beta testing of the system. Once the defect is discovered, it must be analyzed to find out what type of defect it is and why it has occurred (in order to remove a defect we must know the source and type of defect). After analysis of the defect it should be prioritized which determines its level of severity and impact on the system if it is not removed. Then comes the most important step which

is defect removal, a number methods can be used to remove a defect. These methods are discussed in detail further in the paper. Thus, the final step is to verify whether the defect is successfully removed or not. This can be done by technical experts or by the user.
These steps of DMP must be carried out at each level of development to improve the quality of the system and to avoid defects at the time of implementation. Early management of defects results in better development of the software and produces the product within time and budget.

II. Literature Review

Vashisht et al., [1] explains that defects can be found at any stage of SDLC. Existing defects in the software are source cause of system failure and customer frustration. In this paper the author has proposed a noble framework that has wide usability. A framework with neural network has been proposed because recently neural networks used in most application like car automation, automating facial recognition and dynamic modeling of software systems. The framework also offers an interface to promote the system's customer interaction. The system will be able to categorize the defects as high, low and mid-levels. 45 software projects have been used to train the neural network and 15 software projects are used to test the resultant system. And the system has shown 90A system with convolution neural network has shown 12Various reasons behind software failures. Author suggests that failures can be mainly found in requirements, design, coding and knowledge expertise. The author has described SQA as control process that prevents the defect occurrence from the start of the project. [4]
R. Anand et al., [5] suggests that the defects are obvious to be made in any system because of human nature. These defects can be identified with reviews and their count is directly related to the amount of rework to be done. The author further has mapped the defects types on SDLC. For the framework validation data has been collected from 21 different projects and 8 defect types were recognized. For example, inappropriate coding standards, incomplete requirements, inaccurate interface and deviation from tests cases were main defects. ANOVA technique is used to find the comparison between fisher's and turkey's method to finally formulize a grouping method. This introduced a life cycle approach that can be used by companies to handle errors in software development programs. The template can be used in conjunction with the current market-driven defect management tool. [6]
The paper addressed various software quality issues and alternative approaches to improve the quality of the information systems. Performance systems such as CMMI and ISO are used primarily for large-scale projects and not in small-scale projects. Techniques like dive and conquer, automation, extreme programming, FAMI, incremental strategy, audit and verification are discussed. Testing management is process which consists of various activities. For example, test planning, test cases design, test execution and test reporting and tracking. [7] [8]
Qazi et al., [9] briefly describes the existing techniques for software inspection. As inspection can be used to remove defects from systems to improve the quality factors. Formal and informal inspections can be conducted to maximize the performance. Pair programing in agile development are also inspection tools. Author has compared FAGAN method, structured walkthroughs, formal methods and Bayesian Belief Model. Evaluation arrangement with reviewed configuration such as advanced matrices, new testing dimensions and progressive class spaces are used to classify accuracy of the systems. [10]
Y. Kamei et al., [11] quality assurance is a major challenge since the beginning of the software industry. It has therefore received an immense amount of attention and research. This paper is presented to acknowledge the successes in the field of defect prediction and future trends. Developing cross platform defect prediction models, using machine learning tools and transparency are some of the future trends.
Trends of software development paradigms are constantly changing. Performing a comprehensive survey of technology practitioners to find out about these activities could identify trends in software testing. Results from surveys may also allow engineers to understand the relationship between areas such as software testing and reliability. They meant to improve the limitations of information and to find best practices that could then be propagated. [12] The function discovery methods are also discussed for each classifier. Because classifiers can subsequently find differ defects, in conjunction with general sub-sets, certain classifiers can yield better results. [13]
S. Patil et al., [14] automation for defect management system consequently increase the rate of categorization of defect reports. But a big limitation for the systems is the requirement of labelled data for defect. To overcome this problem the author proposes to use explicit sematic analysis in concept-based environment to classify defect reports. In the Wikipedia definition space, the system will quantify comparisons between the defect labels and a defect explanation and then allocate the defect label with the highest resemblance to that defect description. [15]
Q. Song et al., [16] describes that a commonly observed issue is that there are only a few faulty parts (usually referred to as good cases) and a significant number of perfect components (bad cases) in the real-world computer fault data. This variation in data is known as class imbalance. These class imbalance plays an important role in software detect predation system using intelligence. In the proposed experimental evolution, the basis for problems are discussed in accordance with the influence of unbalanced training and its correlations with information mismatch, category form, and the unbalanced process of learning. The value of mismatched good and bad groups is calculated by the consistent coefficient. [17]
Zhendong et al., [18] suggest an automated defect management system using mobile computing and big data techniques. Recently, the traditional fault detection system errs because of incompatibilities. Author has mentioned three defect management obstructs that cause more disturbances. First is to

consume less period to support practitioners correctly place deficiencies. Second is to avoid strong technical reliance. And third is to Establish an advanced training structure for project managers to gain understanding. The intelligent framework proposed works by feeding all the defects in the mobile sets from where the data will be stored on encrypted clouds, the data can also be shared and updated by the management. The system be able to file the defects and analyze them before and after their removal.

III. Defect Management Methodologies

A. Defect Detection and Prevention

Defect detection and prevention is a technique which is used in agile software development. Quality Assurance does not only depends on detection of defect but it also depends on prevention of defect that the defect should not occur once it is detected. This relation can be explained by the following equation:-

$$QA = DefectDetection + DefectPrevention \qquad (1)$$

Detection and prevention are two separate processes, but they work in parallel to ensure efficient quality assurance. In this process, the tester detects a flaw and seeks the root of the defect. Therefore, the defect is removed from the root which prevents it from happening again.

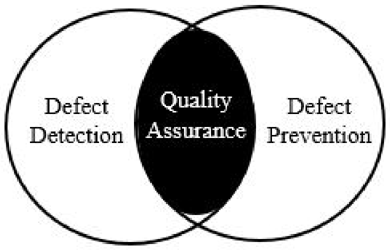

Fig. 2. Graphical Representation of Quality Assurance through Defect Detection and Prevention

a) Detection and Prevention Techniques: Different defect detection and prevention methods can be used by quality engineers, some of them are mentioned below:-

- Walkthrough:
  It is a method in which the developed software is compared with the prototype of the system to ensure the required functionality.
- Defect Logging:
  It is a process in which defects are detected in software through user feedbacks and new fixed versions are generated as output.
- Root Cause Analysis:
  This method is used to analyze the root cause of defects using two approaches i.e Pareto Analysis and Fishbone Analysis.

b) Advantages:

- This technique is helpful in improving the software process but it does not provide any steps to reduce the level of defects in the software.
- It provides efficient cost and schedule saving.
- It saves the development team from rework and scrape.
- It increases the reliability of the system.
- It plays a vital role in enhancing customer satisfaction and trust.

c) Disadvantages:

- It does not provide any steps to reduce the level of defects in the software.
- More resources and time is required at each level of development to perform defect detection and prevention.
- Defect Prevention may take more time than the actual development of the system.

B. Refactoring

Refactoring is one of the most common techniques used in software testing to improve the quality of a software. It basically deals with the restructuring of the code of the software to improve its internal structure.
Refactoring changes or improves the code software system in such a way that it does not effects the behavior of the system. It renders the internal working of the software without causing any change in its external working. In other words we can say that it is mostly used to modify and optimize the code without changing its functionality.

a) Refactoring Techniques: Following are some of methods used for refactoring:-

- Increasing Abstraction
- Breaking the code into logical modules
- Improve terminologies used in the code

b) Advantages:

- Refactoring is an excellent approach to achieve a less complex and understandable code.
- It is easier to identify bugs in the code.
- It improves the design of the software.
- It enhances the readability of the code.

c) Disadvantages:

- It is difficult to perform refactoring if you are running ahead of schedule.
- It may require additional cost.
- Refactoring may introduce bugs into a stable system if done by an inexperienced or incompetent resource.

C. Inspection

Inspections is one of the most common techniques used to detect and manage defects in software project especially the projects using the agile development model.

Inspection is a practice in software engineering to review the work product. The review may be performed by a single tester or by various groups to identify and pinpoint any defects present in the work product. Different work product are inspected including SRS and test plans. Inspection is called the most formal reviews which is done during static testing phase.

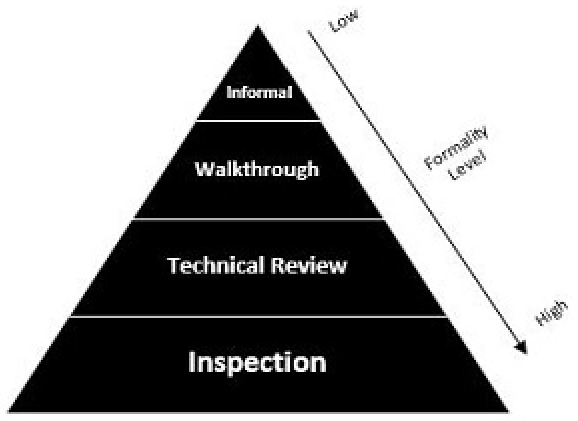

Fig. 3. Formality Levels of different reviews

a) Goals: Following goals are achieved by inspection.

- It assist the team to increase the quality of the inspected document
- Efficient removal of defect can be ensured
- Quality of the deliverables are enhanced
- It helps to perform defect prevention by determining the cause and type of defects found.
- It not only identify the defects but also aids in process improvement.

b) Inspection Process: Following are the stages of inspection process as shown in figure below:

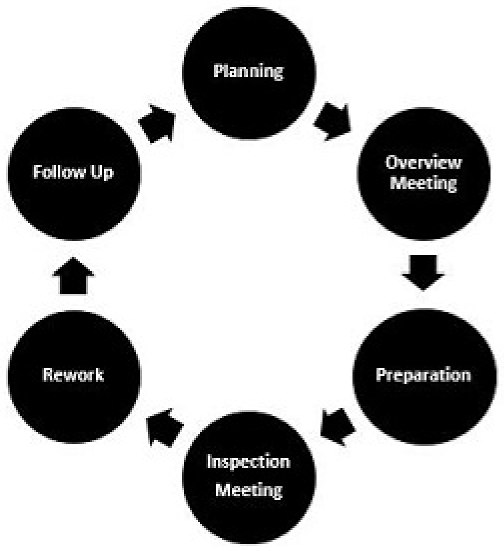

Fig. 4. Processes of Inspection

- Planning: The process of inspection is planned by the moderator.
- Overview meeting: In this meeting, all the related work is explained by the author about the deliverable.
- Preparation: Each inspector examines the work product to identify possible defects.
- Inspection meeting: During this meeting the reader reads through the work product, part by part and the inspectors point out the defects for every part.
- Rework: The author makes changes to the work product according to the action plans from the inspection meeting.
- Follow-up: The changes by the author are checked to make sure everything is correct.

Any of the above stage can be iterated to complete the inspection. Some important aspects should be considered while performing the inspections i.e. having an entry criteria to determine the start of the process and to avoid any incomplete product to penetrate the process. It can be done by using a checklist. An exit criteria should also be present and the completion of the process will be reflected by it.

c) Inspection Roles: During an inspection the following roles are used.

- Author: The person who created the work product being inspected.
- Moderator: This is the leader of the inspection. The moderator plans the inspection and coordinates it.
- Reader: The person reading through the documents, one item at a time. The other inspectors then point out defects.
- Recorder/Scribe: The person that documents the defects that are found during the inspection.
- Inspector: The person that examines the work product to identify possible defects.

d) Advantages:

- Improvement in quality of deliverables resulting in higher product quality
- Helpful in removing defects at early stage
- Find errors early, closer to their source.
- Several errors can be identified at once. With testing errors usually are discovered and fixed sequentially.
- Provide the side benefit of cross-training. Inspections are an opportunity for new developers to learn good (or at least different) programming styles and to learn more about the system under development.
- Schedule deadlines
- Increase customer satisfaction
- Speed up training processes for new products
- Improve development process model
- Team building environment

e) Disadvantages:

- It checks the conformance of product to the specification and not to the real needs of stakeholders, i.e. it is a form of verification.
- They take time and require discipline.

D. Static Analysis

Static Testing is an application's validation or analysis by analyzing the code before executing the application. This analysis methodology aims to analyze internet and non-web applications as well as to expose errors by advanced modeling that may not occur until days, months or years after publication. Static analysis may also be carried out by a professional who checks the software to insure that correct programming principles or protocols are used to create the system. This is often referred to as Code Review and is done by a peer developer, other than the developer who wrote the code.

a) Uses:

- It can provide valuable information for systems documentation.
- It can reduce algorithm processing time
- It can analyze various parts of the programs written by different people to detect errors.
- Maintenance can be helpful.
- It can also generate the programs ' structure graphs.

b) Type of defects:

- A variable with an undetermined value.
- Inconsistency of interface between modules and components.
- Variables that have been declared but never used.
- Dead Code.
- Infringement of programming standards.
- Vulnerability of security.
- Violations of syntax.

c) Roles of tester: Test teams in static analysis need to be much more experienced in fields such as security systems, configuration and coding. The benefits of static analysis techniques should be well known to researchers. Testers should collaborate with other partners to gain knowledge in this field to get benefit producing a higher quality and safety target. Testers need to focus on quality, which involves integrating quality and safety into the process of software development. Test teams can also help to establish fair acceptance criteria and work in collaboration with design and management teams to create the complete flow to enable the tool's successful usage. Static test teams follow a standard document review process, including:

- Standard specification
- Data format planning
- Verification of the material list
- Internally referenced source verification
- Externally cited reference validation
- Screening and review of documents
- Report examination

d) Advantages:

- It can detect flaws at the exact location in the code.
- It can be achieved by qualified developers in computer assurance who fully understand the code.
- Source code can be easily understood by other and potential programmers
- Enables a faster turnaround to corrections and weaknesses to be found earlier in the life cycle of growth, reducing the cost of repairing.
- Few faults were found in later trials
- Dynamic checks measure particular flaws that cannot or hardly be observed
  - Unreachable code
  - Using undeclared variables
  - Uncalled methods
  - Violation of boundary values

e) Disadvantages:

- Once done manually, it is time consuming.
- Automated systems give rise to false positive or false negatives.
- There is not enough trained staff to carry out a thorough analysis of static code.
- Automated systems can give a false sense of protection when it comes to addressing anything.
- The vulnerabilities introduced in the run-time environment are not found

E. Defect Tracking

In software engineering, defect tracking is the method of documenting the recorded faults in a product from start to finish (by review, screening, and customer feedback recording) or making new product versions that correct the defects. In software engineering, defect tracking is essential because complex software structures usually have tens or hundreds or thousands of deficiencies: It is a difficult task to assess and prioritize these vulnerabilities. If the number of defects becomes quite large and the defects need to be tracked over long periods of time, the use of a defect tracking system can make the task of management much easier.

a) Defect Tracking Process: Following goals are achieved by inspection.

The following points explain how defect detection can be done.Analysis: A tester has to be a keen observer when evaluating an application and should also have a lot of expertise in this respect. A tester is expected to apply experimental observation so that the test scenarios can be detected, both from a positive and negative perspective. Defect Entry: If an unexpected system behavior that turns out to be a defect occurs to the tester, it should be entered by the tester in the defect tracking tool.. Recurrence: How to detect and treat a recurrent flaw, whether the defect needs to be recognized as a new defect or whether the previous defect needs to be reopened. Closure: Finally the tester must remove the

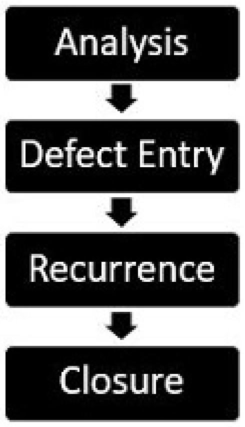

Fig. 5. Defect Tracking Process

defect after addressing all the relevant issues.

b) Defect Tracking Parameters: Defects are tracked based on different criteria such as:

- Defect-Id
- Importance
- Difficulty
- Created by
- Date of Creation
- Appointed to
- Resolved Date
- Resolved By
- Status

c) Advantages:

- When it comes to tracking errors, there is no shortage of tools. Tools for tracking non-technical and technical issues are available.
- Defect tracking helps you make sure that the bugs found in the system are actually fixed.
- Defect tracking tools not only provide a means of ensuring follow-up, but also provide valuable metrics.
- The team can tie defects to changed code, tests, or other data that will enable traceability or analysis of defect trends, depending on the tool being used.
- Defect monitoring systems allow a documentation repository to provide utility to troubleshooters and later support staff if there is a problem solution

d) Disadvantages:

- It takes too long to describe problems in it
- Sometimes the process overhead required to open a bug is more time-consuming than simply fixing the bug.
- With time, the lists of bugs get so long that no one can deal with them anymore, taking up a lot of our time.
- With a huge list of bugs and their fixation, there will not be enough ROI.

IV. PROPOSED DEFECT MANAGEMENT MODEL

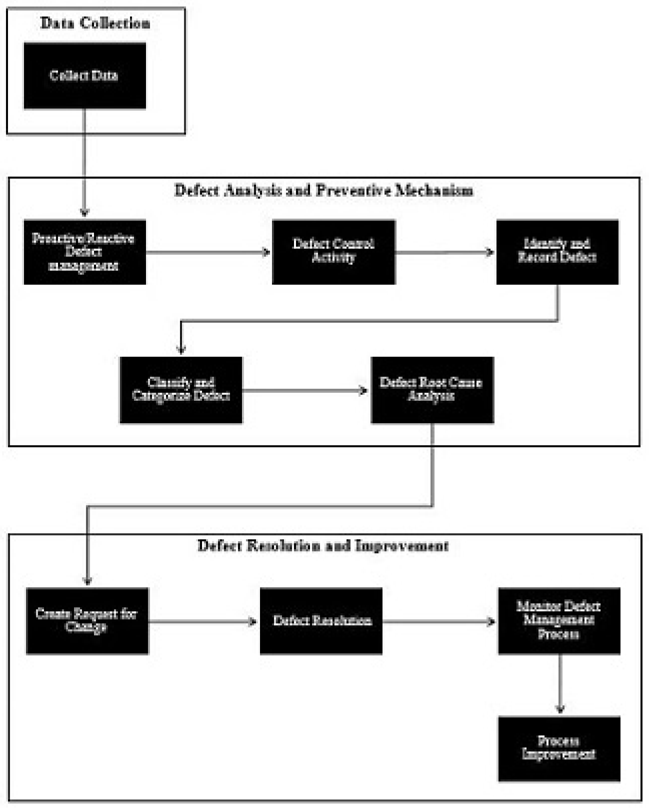
Data Collection
Collect Data
Defect Analysis and Preventive Mechanism
Proactive/Reactive Defect management
Defect Control Activity
Identify and Record Defect
Classify and Categorize Defect
Defect Root Cause Analysis
Defect Resolution and Improvement
Create Request for Change
Defect Resolution
Monitor Defect Management Process
Process Improvement

Fig. 6. Processes of Inspection

A. *Collect Data*

In order to accomplish a better first time defect repair rate, IT departments must collect data from the help desk, QA and project team leaders. The aim of gathering the defect information from help desk, QA and project teams is to fix most defects as soon as possible before shipping the product to the actual customer.

B. *Proactive/Reactive Defect management*

best approach is to remove the defects as they have been discovered. This can only be done if the company employs defect-prevention strategies and procedures. The goal of prevention of defects is to completely eliminate the defects so that they will not be able to reoccur in the future. Proactive fault management's primary goal is to detect and fix identified flaws as early as possible before any major issues arise. Once the defects have been found, try to remove each and every defect. Try to reduce the effect of defects that can not be removed. The focus of the reactive flaw management process is to identify the root causes of the reported problem.

C. *Defect Control Activity*

Defect management operation starts when the defect data analysis shows repeated problems or the defects analyzed does not meet any of the problems that occur.

D. Identify and Record Defect

If a defect indicates recurring problems all defects are detected and reported in the defect management system. The record of defects must be connected to the incident or problem. This will contribute towards the identification of the defect solution or future work. A defect has no significance until the defect is reported, and the developer must also recognize that the defect is found to be valid.

E. Classify and Categorize Defect

Various classification systems are used to classify defects such as Orthogonal Defect Classification (ODC) and Root Cause Analysis (RCA). Defects are categorized according to classification, importance, severity and impact For example, the possible categories of software defect can be functional, interface and algorithm, etc. The impact of the defect on the company and priority depends on the urgency and impact of the defect.

F. Defect Root Cause Analysis

Investigate and identify the underlying causes of the defect after classifying and categorizing defects. For the root cause analysis, various defect analysis techniques, methods and standard processes are used.

G. Create Request for Change

Create a change request for the development team to introduce a permanent solution for the defect found.

H. Defect Resolution

The resolution process starts once the developer acknowledges that the found defect is valid. The developer should be aware of the importance of correcting a defect before addressing the defect. The developer should notify all related parties about the defect status after the defects have been resolved.

I. Monitor Defect Management Process

Project management should track the process of defect management on a continuous basis. The progress of the defect resolution process and the impact of defects on customers should be reported by project management. Monitoring should be carried out on the basis of the actual requirements set out in the requirement specification document.

J. Process Improvement

Many companies have skipped this process, although this system is one of the main payback components. Participants should go back to the stage of the source of the defect and analyze what caused the defect. The validation process in which the defect should be caught earlier must be reviewed after that. This measure will not only strengthen the review process but also enhance the capacity of the client for organizational business logic.

V. Conclusion

One of the success measures for delivering a quality software program is a well-established process for the control of defects. Three stages of fault identification, evaluation of faults and avoidance of defects are used to control code defects. The current proposal is easy to use and reinforces the administration and screening process of organizational defects. This policy has assigned accountability for each form of defect to a particular test stage, making for a more efficient review process by eliminating repetition and reducing costs. It helped the testers to focus only on the vulnerabilities they were given. The significant improvement of this analysis is to set up an organization's fault control system template to limit the number of errors and deliver a software product of value.

VI. Future Work

The goal of quality assurance is to have mechanism to make a legitimate estimation of the categories of faults in the critical objects produced during the life-cycle of the code. Experts have to analyze the factors affecting the emergence of defects in further specifics and to indicate those factors in the categorization of defects. Furthermore, in order to assist the compilation of scientific methodology, they have to spend more time in establishing relevant yet systematic identification of defects.

References

[1] Vashisht, V., Lal, M. and Sureshchandar, G. (2015) A Framework for Software Defect Prediction Using Neural Networks. Journal of Software Engineering and Applications, 8, 384-394. doi: 10.4236/jsea.2015.88038.

[2] J. Li, P. He, J. Zhu and M. R. Lyu, ”Software Defect Prediction via Convolutional Neural Network,” 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, 2017, pp. 318-328. doi: 10.1109/QRS.2017.42

[3] Danilov, A & Samotsvet, D & Mugatina, V. (2019). Using neural network models in the quality management system for the software defect prediction. IOP Conference Series: Materials Science and Engineering. 537. 042038. 10.1088/1757-899X/537/4/042038.

[4] Y. Chen, J. Chen, Y. Gao, D. Chen and Y. Tang, ”Research on Software Failure Analysis and Quality Management Model,” 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, 2018, pp. 94-99. doi: 10.1109/QRS-C.2018.00030

[5] R. Anand, K. David and S. S. Sagayaraj, ”Identifying the impact of defects among the defect types in Software development projects,” 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Chennai, 2015, pp. 146-152. doi: 10.1109/ICSTM.2015.7225404

[6] A. A. Rahman and N. Hasim, ”Defect Management Life Cycle Process for Software Quality Improvement,” 2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS), Kota Kinabalu, 2015, pp. 241-244. doi: 10.1109/AIMS.2015.47

[7] Rashid, Junaid & Nisar, Muhammad. (2016). How to Improve a Software Quality Assurance in Software Development-A Survey.

[8] S. Kukreja, A. Singhal and A. Bansal, ”A critical survey on test management in IT projects,” International Conference on Computing, Communication & Automation, Noida, 2015, pp. 791-796. doi: 10.1109/CCAA.2015.7148481

[9] Qazi, Asad & Shahzadi, Sidra & Humayun, Mamoona. (2016). A Comparative Study of Software Inspection Techniques for Quality Perspective. International Journal of Modern Education and Computer Science (IJMECS). 10. 9-16. 10.5815/ijmecs.2016.10.02.

[10] Li, Libo & Lessmann, Stefan & Baesens, Bart. (2019). Evaluating Software Defect Prediction Performance: An Updated Benchmarking Study. SSRN Electronic Journal. 10.2139/ssrn.3312070.

[11] Y. Kamei and E. Shihab, ”Defect Prediction: Accomplishments and Future Challenges,” 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, 2016, pp. 33-45. doi: 10.1109/SANER.2016.56

[12] M. Kassab, J. DeFranco and P. Laplante, ”Software Testing: The State of the Practice” in IEEE Software, vol. 34, no. 05, pp. 46-52, 2017. doi: 10.1109/MS.2017.3571582

[13] Bowes, D., Hall, T., & Petrić, J. (2017). Software defect prediction: do different classifiers find the same defects? Software Quality Journal, 26(2), 525–552. doi:10.1007/s11219-016-9353-3

[14] S. Patil, ”Concept-Based Classification of Software Defect Reports,” 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, 2017, pp. 182-186. doi: 10.1109/MSR.2017.20

[15] Ozakinci R., Tarhan A. (2016) The Role of Process in Early Software Defect Prediction: Methods, Attributes and Metrics. In: Clarke P., O’Connor R., Rout T., Dorling A. (eds) Software Process Improvement and Capability Determination. SPICE 2016. Communications in Computer and Information Science, vol 609. Springer, Cham

[16] Q. Song, Y. Guo and M. Shepperd, ”A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction,” in IEEE Transactions on Software Engineering. doi: 10.1109/TSE.2018.2836442

[17] Arar, Ö. F., & Ayan, K. (2015). Software defect prediction using cost-sensitive neural network. Applied Soft Computing, 33, 263–277. doi:10.1016/j.asoc.2015.04.045

[18] Zhendong, Li & Wei, Song & Chong, Wang & Wenbo, Yu & Yanping, Diao & Wei, Zhao & Qimeng, Liu. (2016). Study on intelligent defect management strategy based on cloud mode. 51-54. 10.1109/ITNEC.2016.7560317.