I.J. Modern Education and Computer Science, **2016, 10, 9-16**
Published Online October 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2016.10.02

A Comparative Study of Software Inspection Techniques for Quality Perspective

Asad Masood Qazi
University Institute of Information Technology – PMAS – Arid Agriculture University Rawalpindi, Pakistan
Email: asadmasood1@hotmail.com

Sidra Shahzadi and Dr. Mamoona Humayun
University Institute of Information Technology – PMAS – Arid Agriculture University Rawalpindi, Pakistan
Email: {sidra.shahzadi363@gmail.com, mamoona@uaar.edu.pk}

Abstract—Software inspection is useful to detect the defects in any stage software development methodology especially in early stages. Inspection of software defects can improve the software product quality by decreasing rework cost and time from documents, code, and other deliverables. The objective of this study is to identify existing software inspection techniques which help practitioners and software engineers to improve the software quality and to compare them according to some quality attributes. Rather than proposing new techniques we focus on synthesize the existing approaches. A comparison of some approaches is conducted and analyzed which approach is more feasible for the detection of defects. The results of this study show that there are many formal and informal techniques available in literature related software inspection, it is difficult to say well to one of them, but our analysis focused on finding such techniques which cover maximum quality factors in an inspection. Software inspection is an umbrella activity of whole software development life cycle and we found different approaches and frameworks in software inspection which can full fill our desired parameters to improve software quality.

Index Terms—Software Inspection, Software Quality Assurance, Software Testing, Software Defects.

I. INTRODUCTION

Inspection refers to examine the software systematically intend to detect the defects. The inspection also refers to ensure that; a product which is developed is same as described in documents. Software inspection is a good approach to detection of defects in all stages of software development [1]. The concept of software inspection arose approximately 35 to 40 years ago. In the start, it was considered that; Software inspection doesn't involve coding to detects the defects, inspection of software can be performed through use cases models, checklists etc. But later on some authors also uses some inspection techniques in object oriented programming and software coding as well. [2] Software inspection is also important because it bridges a gap between software testing and software formal verification. As, software testing has major concerns with the software industry, whereas formal verification is related to the academic side, so software inspection lies between testing and formal verification. There is no standard neither given in literature nor adopted by any software organization of software inspection. However, in many existing software engineering process models, software industries case studies and in academia different terminologies are used which are ultimately affected by software inspection. For example pair programming in Extreme Programming (XP). [3]

Software inspection is somehow part of some of the software processes indirectly. For example, we can see in the XP (Extreme Programming) of Agile based software development methodologies, pair programming concept is defined, in which one group member writes code, design or another document and one is dedicated to the review on runtime which supports the knowledge sharing as well as inspection.

Software inspection is not widely used in organization these days. Inspection refers to peer reviews, walkthroughs, and structured reviews. There are many reasons for not using inspection techniques widely in software organizations which are actually myths and these are highlighted by Radice [4]:

(1) Inspection is one way technique
(2) Inspections are not easy to do
(3) Inspections add an extra cost in software quality phase

The reasons of such myths are evolved, because the inspection was considered as a process, having low technology involvement so it was not an enjoyable or interesting task for software engineers or inspectors.

A. *Traditional Software Inspection Technique Process*

The traditional process of Inspection is shown in **0** below. [4] Adaptation of formal software inspection plays an important role in ensuring software quality. Traditional approaches mainly involve informal techniques using walkthroughs, checklists etc. and the meetings are arranged at author's or programmer's desks and review process is held usually in an informal way. Walkthroughs can be structured as well as semi-structured. Traditional Inspection approach can be applied to any software artifact like document or code.

Software inspections may involve software testing, but not necessary because inspection is a review based activity. Inspection process helps in knowledge sharing with in organization because of review meetings, where two or more partners come together to review the work of an individual or team.

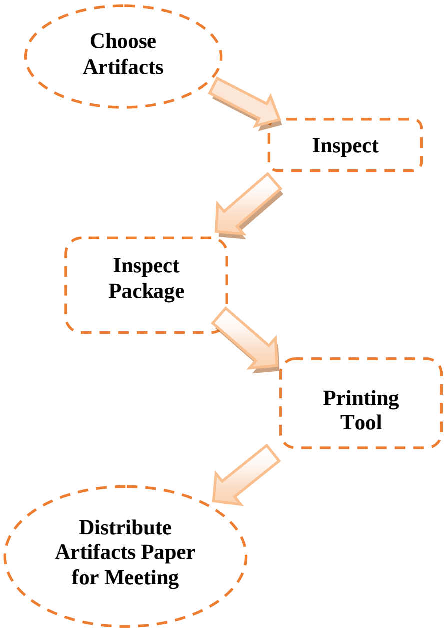

Fig.1. Flow of Software Inspection Process Traditional Approach

The remainder of this paper is structured as follows: Section 2 summarizes some of the related work. Section 3 describes the existing defect detection techniques and the comparative analysis methodology, used in this paper. Section 4 provides comparison of existing software tools which supports automatic software inspection. Section 5 provides a result analysis of Section 3 and 4. Section 5 presents the future work and also concludes the work.

II. Related Work

Software inspection is being used for more than 35 years in research area as well as in software organizations. The main purpose of software inspection is to identify the software defects in early stages to overcome the complexity, budget and to improve software quality.

Taba et.al, (2012) [4] proposes a Scenario based software inspection method and compare it with the traditional approaches. The proposed model is a formal technique and full fills many quality attributes in software inspection process. It also involves pre and post activities. The study lacks in comparative analysis when it only targets on large scale software organizations. Because inspection itself is an approach, which usually can only be taken in large scale organizations.

Taba et.al, (2012) [5] proposes another software inspection model named as DAMEO (Defect Management Oriented Inspection), which is again a complete formal approach and can only be successfully applied in large scale software organizations. It increases the efficiency by improving execution time and effectiveness in software inspection process while comparing with traditional approaches. Three parameters are been taken in this comparative study which are FD (Founded Defects), RD (Remaining Defects) and FT (False Positive). Under these parameters DAMEO gives effective results over traditional approaches.

Nancy. S et.al, (2002) [6] tailored the Fagan methodology of software inspection and compare the results with an experiment approach. The Fagan methodology is reduced in four steps rather than six to seven steps. The proposed steps are Planning, Study, and Meetings and follow up. The proposed approach reduces the number of hours spends on preparation phase of individuals which increases the effectiveness and productivity as well as overall time and cost. The results of comparative analysis lack because of involving a small number of parameters.

According to Guilherme (2014) [7], there are many pieces of evidence about the feasibility and efficiency of applying software inspection techniques. Software inspection is a pre-test activity and it is also an important activity of verification validation and testing (VV&T) activities of software development. Software inspection can be applied to any artifact. HP uses inspection techniques, code, testing and documentation. IBM uses inspection for Design and Code section. ICL (an operating system) uses inspection at design level. Shell Research use inspection in Requirements phase.

Porter et.al, (1996) [8] in their review study of software inspection techniques, compare existing methodologies. A comparative analysis presents on the basis of local as well as global analysis. The local analysis doesn't affects software development process during an inspection process. The parameters of comparison are the num number of team size, number of sessions, collection technique, defect detection method and feedback as post development or post inspection activity. The study is presented many years ago, so approaches which are presented later later should also need to synthesize.

A. Aurum et.al, (2002) [9] covers a review of 25 years work of software inspection. Software inspection formally introduced by Fagan (1976) [10] and the later methodologies actually improve the work of Fagan. A comparative analysis of 25 years of work has been presented in this study. Upto 2002 (the year of this publication) there were following advancements in software inspection area: Fagan's Inspection (1976) [10], Active Design Review (1985) [11], Two-person Inspection (1989) [12], N Fold Inspection (1990) [13], Phase Inspection (1993) [14], Inspection without Meetings (1993) [15], Gilb Inspection (1993) [16]. So, there is need to synthesize the data of software inspection methodologies up to current work.

F Macdona et.al, (1995) [17] presents a review of existing tools which supports the software inspection process. Tools can supports software inspection process in different ways like Documentation support, Annotation support, Automatic defect detection, Checklists, Enforcement, Meetings support, Decision support, Metrics collection, Code inspection (either statement by statement or complete code review). Some of the tools are ICICLE [18], Scrutiny [19], CSI [20], InspeQ [21], and CSRS [22]. The study covers the review of inspection tools up to 1995, so there is need to incorporate the tools which developed later to this review study.

Souza et.al, (2013) [23] in their research includes six software engineers and uses Fagan's and Gilb's inspection methodology with the roles of moderator, reviewer, author, and reader. The aim of this study is to inspect the product line process in the software organization. A result shows that incompleteness, ambiguity, incorrectness, unnecessary information, inconsistency, and non-traceability were found in industrial product line projects using these software inspection techniques. The reason to add this study to related work is a merger of Fagan's and Gilb's methodology to improve the results of software inspection process.

Elberzhager et.al, (2014) [24] compares the software inspection process with software testing. Software inspection is primarily a review process to detect the defects data just like in quality monitoring and testing is done on the output of software inspection which has defected data / defected product. Comparison of inspection and testing is important to include because there are some inspection techniques which includes the testing itself, and some of the techniques are used just to identify the defected data or defected part of a system and shifted forward for the testing process. Authors of [25] also support this argument that inspection process is not a replacement for testing. And to inspect the software deliverables, an inspector should also belong to same environment or organization.

In [26] and [27] the role of software inspection in software industries of Pakistan and Srilanka respectively. In Pakistan's perspective, software inspection phase is analyzed using ETVX (Entry, Tasks, Validation, and Exit) model and shows that 75 to 100 projects becomes successful using software inspection, whereas without using software inspection, the success ratio of software projects are limited to 50 to 75. In the Sri Lankan software, industry inspection is also formally implemented and industries have proper roles of software inspection process and getting following benefits from software inspections: reduce overall effort, increase productivity and reduces cost. In both of these references, formal inspection is used rather than informal or traditional approach of software inspection techniques.

Kollanus et.al, (2006) [28] argues that software inspection is important in software engineering disciplines but these are not actually implemented properly in some of the organizations. Data in this study is gathered from six software organizations and find the problems or hurdles in the way of inspection. Authors found that there is the lack of inspection training, limited formality with inspection process and immaturity of inspection measurement techniques in most of the software organizations. We include this case study, to identify the new or modified approaches to software inspection from existing literature, which may reduce these obstacles from the inspection process.

III. Comparative Analysis of Software Inspection Techniques

A. Parameters of Comparison

(1) Pre Inspection criteria
(2) Defects Detection
(3) Defects Removal
(4) Efficiency & Effectiveness
(5) Complexity
(6) Post Inspection procedure

There are some reasons for taking these parameters for comparison. All the existing software inspection methodologies are focuses on before or pre-inspection steps which include the preparation etc., then actual inspection is being done and finally, the post-inspection steps which may include the implementation of reviews and measurements of effectiveness or efficiency etc.

B. Formal and Model Based Approaches

Traditional approaches mainly focused on informal or semi-formal of software inspection approaches. Inspectors use checklists before informal inspection meetings, and there were informal reviews and some structured walkthroughs to inspect the elements of software. However, these approaches may contribute the results in some way as studies shows that if the error in requirements does not correct in early stages, the cost may exceed up to 40 percent of actual cost [29] [30] [31]. And another study argues that inspection should be done during design and analysis phase, to detect the defects and then it will decrease the cost from 10 to 100 times less than the testing phase. [32]

Some of the models and formal techniques can be found from different sources of literature.

(1) Fagan Methodology
(2) Glib Methodology
(3) Phased Inspection
(4) Scenario Based Inspection Method
(5) DEMAO (Defect Management Oriented Inspection)

C. Fagan Methodology

FAGAN methodology [10] is a complete software inspection methodology and proposes proper sequence of steps and roles. Steps are: Pre and Post inspection activities, inspection meetings and the roles are software inspector, software tester, and moderator. Fagan's methodology consists of six phases planning, overview, preparation, examination, rework and follow-up. Firstly moderator at planning phase identifies inspector's role, distribution and verification of inspection material etc. Secondly, an overview is done by the author. Thirdly inspector prepares for meeting and role and it will also improve the process if this preparation inadequate then moderator cancels it. Finally, an Author will correct all defects which will be verified by the moderator.

D. Glib Methodology

Glib inspection methodology [16] was developed by the Tom Glib in 1993, like Fagan methodology there are six phases planning, overview, preparation, examination, rework, follow-up etc. according to Glib inspection methodology which is used by an organization is depend on its type of business , it's up to the customer's choice whether he choose formal or semi-formal inspection process. An extra step was added by Gilb to inspection process for the improvement of software development process that will produce the document which will be inspected by the inspector.

E. Phased Inspection

Phase inspection [14] was proposed by Knight & Myer (1993) where software products are inspected in series of steps called phases where each phase has the specific objective. At each phase product is examine, validate for specific properties of a product. We cannot move forward until corrections are completed. There are two types of phase single-inspector, multiple-inspector. A single inspector uses a checklist that must satisfy the by each product. Multiple-inspector phases are designed for such properties that cannot be described through binary questions. In phase inspection, individual checking is also done via meeting called reconciliation that provides various opportunities for fault detection.

F. Scenario Based Inspection Model

Literature shows that for the removal of defects various testing models, automated and manual tools had been proposed, but still they are failed. Most of the software inspection model, techniques focuses only on artifacts but, the proposed model provides an inspection technique that removes some possible defects in all phases of software development. It does focus not only each phase of SDLC but it also concentrate on documents, deliverable working products and conduct inspection process implicitly and gradually. Defect removal, determination, and defect learning are three golden steps, where defect learning is an interesting point basic factor of a scenario-based model. This must be intelligent, its learning plan creates, executes according to founded results. A case study was conducted for the evaluation of this model; it is more efficient as compared to other traditional inspection processes. [4]

G. DEMAO (Defect Management Oriented Inspection)

DEMAO [5] was proposed for the improvement of software quality, generally, it focuses on inspection process inconsistencies. There were four core components of proposed models. Core components of proposed models were (1) defect management, (2) cause and effects, (3) supervision function (4) inspection function. Defect management is an important in any inspection process in DEMAO it was done through relational database together with the knowledge base for maintenance of common defect classification. Finding more defects in less time is a major objective of an inspection process. Cause and effect dependencies can be finding through the Knowledge base. In DEMAO Like traditional software inspection Supervision function is not limited to coordination, it also defines inspection session, develop team charter, approve inspectors' profiles, and arrange meetings for inconsistencies removal .internal and external inspection id done by inspection function. These techniques increase efficiency by decreasing execution time and increase effectiveness by discovering more error and defects. . The most effective features of this model are a reduction in time by providing facilities and formatted documents and disadvantage of was limited on flexibility.

IV. Comparison Table for Commonly Known Improvements in Software Inspection Process

Fagan's Methodology is considered as first and base of formal software inspection methodologies. New methodologies are actually an update of this methodology. A list of some commonly known software inspection methodologies are given below in comparison table, **Table 1**. And another analysis is also presented in **0.** in which a frequency to measure the software quality is given to analyze the result of each software inspection technique.

V. Comparison of Software Inspection Tools

There are number of software tools and IDEs (Integrated Development Environments) [37] available which automatically inspect or review the software code, and indicates the errors, warnings, exceptions etc. some tools inspect statement by statement or line by line of coding, and some tools inspect complete source code. Some tools are also available which measures the complexity of software as well. Some of the common known tools, which support any phase of software inspection, are given in comparison table below in **Table 2.** And a more depth analysis of findings of software inspection tools are given below in form Graphical representation in **0.**

VI. Conclusions and Future Work

In this paper, we have conducted a survey to found existing approaches of software inspection process. We start with the history of traditional software inspection process and moves gradually towards formal software inspection process. We have found that Fagan's methodology is considered as a base of formally based inspection approaches. Later on, we have done a comparative analysis of commonly known software inspection methodologies and models to improve overall software quality. Besides these models, we also include the tools which are used to automate the inspection process. Software tools can support the documents handling, code inspection, meetings, checklists and other related activities of software inspection.

This comparative analysis is the base of our future work to gather the literature data to finding the methodologies which can improve the software quality attributes in all the phases of software development lifecycle.

ACKNOWLEDGEMENT

The authors wish to thank all the faculty members of University Institute of Information Technology – PMAS - Arid Agriculture University Rawalpindi especially Dr. Mamoona Hunayun for their supervision in this work.

Table 1. Comparative Analysis of Software Inspection Models and Techniques

Sr. No	Year of Publication	Name of Technique	Approached Steps	Roles	Evaluation Criteria	Impact on Quality Factor	Reference(s)
1.	1976	Fagan's Inspection	Planning, Overview, Preparation, Inspection Meeting, Rework, Follow-up	Moderator, Author, Reader, Tester	Experiment	Detects the Defects	[33]
2.	1985	Active Design Review	Preparation, Inspection Meetings	Reviewers	Experiment	Reduces Complexity	[33] [11]
3.	1988	Code Inspection Model	Efforts Estimation	Not Defined	Experiment / Case Study	Estimated Density and Effectiveness of Code	[46]
4.	1989	Two - Person Inspection	Planning, Overview, Preparation, Inspection Meeting, Rework, Follow-up	Author, Reviewer	Experiment	Reduce no. of Roles	[12]
5.	1989	Structured Walkthroughs	Checklists, Meetings	Reviewer	Experiment	Completeness	[38] [9] [39]
6.	1990	N - Fold Inspection	Planning, Overview, Preparation, Inspection Meeting, Rework, Follow-up	Author, Reviewers, Moderator	Experiment	Reduce Time of Meetings	[13]
7.	1993	Phased Inspection	Planning, Overview, Preparation, Inspection Meeting, Rework, Follow-up	Inspector, Reviewer	Experiment	Portability, Maintainability, Reusability	[14]
8.	1993	Inspection Without Meetings	Correspondence, Nominal and Depositions	Author, Reviewers	Experiment	Reduce time to face to face meetings	[15]
9.	1993	Gilb's Inspection	Planning, Overview, Preparation, Inspection Meeting, Rework, Follow-up	Author, Reviewers, Moderator	Experiment	Detects the Defects	[16]
10.	2000	Biffl's Re-Inspection Model	Individual Detection, Team Meeting, Defect Correction	Not Defined	Experiment	Improve Product Quality	[47]
11.	2001	Bayesian Belief Model	Semantic Network Model for measuring effectiveness	Moderator	Experiment, Case Study	Increase Effectiveness of Existing Process by reducing no. of roles	[34] [35]
12.	2007	Robust & Flexible	Re-Engineering Process	Not Defined	Academic Projects	Reliable in Re-Engineering Phase	[44]
13.	2012	Scenario Based Inspection Model	Defect Determination, Defect Removal, Defect Learning	Roles required for Analysis and Design Phase	Experiment, Case Study	Improves efficiency and effectiveness	[4]
14.	2012	DEMAO Inspection Model	Develop and Maintain Checklists, Defect Management, Cause and Effect Dependence, Competitive Advantage, Supervision Functions	Trainers, Reviewers, Moderators	Experiment, Case Study	Improves efficiency and effectiveness	[5]
15.	2012	Intelligent Model	Preparation, Defect Plan Design, Generate Inspection Routines, Inspection Process Evaluation	Not Defined	Experiment / Case Study	Effectiveness / Efficiency	[45]

Table 2. Comparative Analysis of Software Inspection Tools

Sr. No	Year of Publication	Name of Tool	Open Source	Documents Handling	Individual Preparation	Meeting Support	Data Collection	Code Review / Manage ment	Reference(s)
1.	1995	ICICLE	NO	YES	YES	YES	YES	YES	[17] [18]
2.	1995	Scrutiny	NO	YES	YES	YES	YES	YES	[17] [19]
3.	1995	CSI	NO	NO	NO	NO	YES	NO	[17] [20]
4.	1995	InspeQ	NO	YES	YES	YES	YES	YES	[17] [21]
5.	1995	CSRS	NO	YES	YES	YES	YES	NO	[17] [22]
6.	2003	Adobe Acrobat	NO	YES	NO	NO	NO	NO	[40] [41]
7.	2003	IBIS	NO	YES	NO	NO	NO	NO	[40] [42]
8.	2004	FlexeLint	NO	NO	YES	NO	NO	YES	[36]
9.	2004	Reasoning's Illuma	NO	NO	YES	NO	NO	YES	[36]
10.	2004	Klocwork	NO	NO	YES	NO	NO	YES	[36]
11.	2004	MINDER	NO	NO	NO	NO	NO	YES	[42]
12.	2005	MS Word	NO	YES	NO	NO	NO	NO	[40]
13.	2005	XATI	NO	YES	NO	NO	NO	NO	[40]

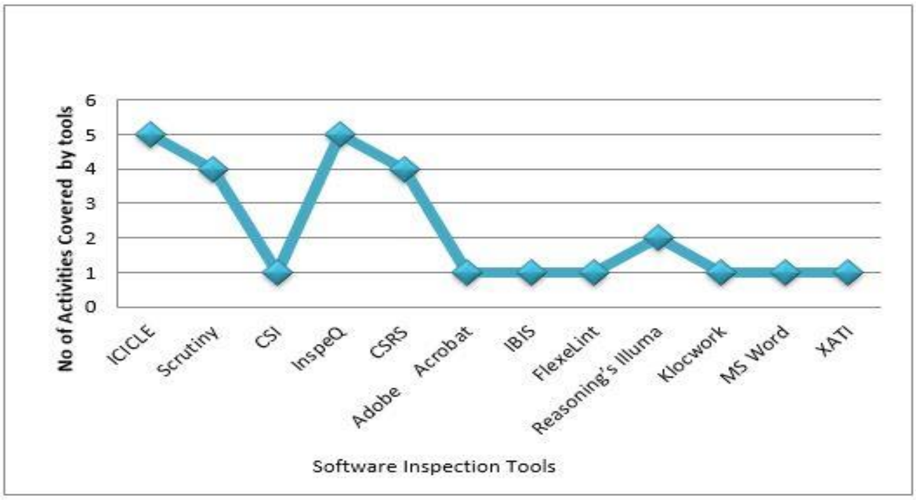

Fig.2. Analysis on Software Inspection Tools with respect to supported features

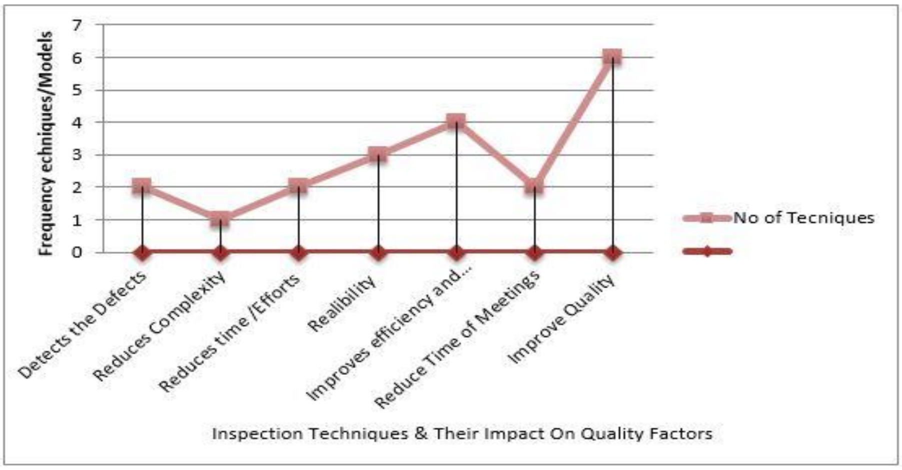

Fig.3. Software Inspection Techniques along frequency of Software Quality Performed

REFERENCES

[1] Parnas, D.L., Lawford, M.: “The role of Inspection in Software Quality Assurance”. IEEE Transactions on Software Engineering 29(8) (2003)

[2] Abrahamsson, Pekka, Michele Marchesi, and Frank Maurer. Agile processes in software engineering and extreme programming. Springer, 2009.

[3] Radice, Ronald A. “High Quality Low Cost Software Inspections”. Andover, Mass.: Paradox icon Publishing, Jan.2002.

[4] Taba, Navid Hashemi, and Siew Hock Ow. "A Scenario-Based Model to Improve the Quality of Software Inspection Process." Computational Intelligence, Modeling and Simulation (CIMSiM), 2012 Fourth International Conference on. IEEE, 2012.

[5] Taba, Navid Hashemi, and Siew Hock Ow. "Improving Software Quality Using a Defect Management-Oriented (DEMAO) Software Inspection Model." 2012 Sixth Asia Modelling Symposium. IEEE, 2012.

[6] Eickelmann, Nancy S., et al. "An empirical study of modifying the Fagan inspection process and the resulting main effects and interaction effects among defects found, effort required, rate of preparation and inspection, number of team members and product 1st pass quality." Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE. IEEE, 2002.

[7] Horta Travassos, Guilherme. "Software Defects: Stay Away from Them. Do Inspections!." Quality of Information and Communications Technology (QUATIC), 2014 9th International Conference on the. IEEE, 2014.

[8] Porter, Adam, Harvey Siy, and Lawrence Votta. "A review of software inspections." Advances in Computers 42 (1996): 39-76.

[9] Aurum, Aybuke, Håkan Petersson, and Claes Wohlin. "State‐of‐the‐art: software inspections after 25 years." Software Testing, Verification and Reliability 12.3 (2002): 133-154.

[10] Fagan, M. E. "Design and code inspections to reduce errors in program development." IBM Journal of Research and Development 15.3 (1976): 182.

[11] Parnas, David L., and David M. Weiss. "Active design reviews: principles and practices." Proceedings of the 8th international conference on Software engineering. IEEE Computer Society Press, 1985.

[12] Bisant, David B., and James R. Lyle. "A two-person inspection method to improve programming productivity." IEEE Transactions on Software Engineering 10 (1989): 1294-1304.

[13] Martin, Johnny, and Wei Tek Tsai. "N-fold inspection: A requirements analysis technique." Communications of the ACM 33.2 (1990): 225-232.

[14] Knight, John C., and E. Myers. "An improved inspection technique." Communications of the ACM 36.11 (1993): 51-61.

[15] Votta Jr, Lawrence G. "Does every inspection need a meeting?" ACM SIGSOFT Software Engineering Notes 18.5 (1993): 107-114.

[16] Gilb, Tom, Dorothy Graham, and Susannah Finzi. Software inspection. Addison-Wesley Longman Publishing Co., Inc., 1993.

[17] Macdona, F., et al. "A review of tool support for software inspection." Computer-Aided Software Engineering, 1995. Proceedings, Seventh International conference.

[18] Sembugamoorthy, V., and L. Brothers. "ICICLE: Intelligent code inspection in a C language environment." Computer Software and Applications Conference, 1990. COMPSAC 90. Proceedings, Fourteenth Annual International. IEEE, 1990.

[19] Gintell, John, et al. "Scrutiny: A collaborative inspection and review system." Software Engineering—ESEC'93. Springer Berlin Heidelberg, 1993. 344-360.

[20] Mashayekhi, Vahid, et al. "Distributed, collaborative software inspection." Software, IEEE 10.5 (1993): 66-75.

[21] Knight, John C., and E. Myers. "An improved inspection technique." Communications of the ACM 36.11 (1993): 51-61.

[22] Johnson, Philip M., and Danu Tjahjono. "CSRS User Guide." (1993).

[23] Souza, Iuri Santos, et al. "Evidence of software inspection on feature specification for software product lines." Journal of Systems and Software 86.5 (2013): 1172-1190

[24] Elberzhager, Frank, Jürgen Münch, and Danilo Assmann. "Analyzing the relationships between inspections and testing to provide a software testing focus." Information and Software Technology 56.7 (2014): 793-806.

[25] Ackerman, A. Frank, Lynne S. Buchwald, and Frank H. Lewski. "Software inspections: an effective verification process." IEEE software 3 (1989): 31-36.

[26] Waqas Ali, Zia-Ur-Rehman, Akhter Badshah, Ali Javed,"Software Inspection in Software Industry: A Pakistan's Perspective", IJMECS, vol.7, no.3, pp.47-53, 2015.

[27] Jayatilake, S. M. D. J. T., et al. "Role of software inspections in the Sri Lankan software development industry." Computer Science & Education (ICCSE), 2013 8th International Conference on. IEEE, 2013.

[28] Kollanus, Sami, and Jussi Koskinen. "Software inspections in practice: Six case studies." Product-Focused Software Process Improvement. Springer Berlin Heidelberg, 2006. 377-382.

[29] O’Regan, Gerard. "Software Inspections Capability Maturity Model Integration." Introduction to Software Quality. Springer International Publishing, 2014. 101-118

[30] O’Regan, Gerard. Introduction to Software Quality. Springer New York, 2002.

[31] Boehm, Barry W. Software engineering economics. Vol. 197. Englewood Cliffs (NJ): Prentice-hall, 1981.

[32] Tyran, Craig K. "A software inspection exercise for the systems analysis and design course." Journal of Information Systems Education 17.3 (2006): 341.

[33] Porter, Adam, Lawrence G. Votta Jr, and Victor R. Basili. "Comparing detection methods for software requirements inspections: A replicated experiment." Software Engineering, IEEE Transactions on 21.6 (1995): 563-575.

[34] Cockram, Trevor. "Gaining confidence in software inspection using a Bayesian belief model." Software Quality Journal 9.1 (2001): 31-42.

[35] Adams, Edward N. "Optimizing preventive service of software products." IBM Journal of Research and Development 28.1 (1984): 2-14.

[36] Nagappan, Nachiappan, et al. "Preliminary results on using static analysis tools for software inspection." Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on. IEEE, 2004.

[37] Parab, Jivan S. Exploring C for microcontrollers: A hands on approach. Springer Science & Business Media, 2007.

[38] Van Emden, Maarten H. "Structured inspections of code." Software Testing, Verification and Reliability 2.3 (1992): 133-153.

[39] Lewis, Clayton, et al. "Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces." Proceedings of the SIGCHI conference on

Human factors in computing systems. ACM, 1990.
[40] Hedberg, Henrik, and Jouni Lappalainen. "A preliminary evaluation of software inspection tools, with the DESMET method." Quality Software, 2005. (QSIC 2005). Fifth International Conference on. IEEE, 2005.
[41] Harjumaa, Lasse. "Distributed software inspections-an experiment with Adobe Acrobat." Proceedings of the IASTED International Conferenceon Computer Science and Technology (2003): 26-31.
[42] Lanubile, Filippo, Teresa Mallardo, and Fabio Calefato. "Tool support for geographically dispersed inspection teams." Software Process: Improvement and Practice 8.4 (2003): 217-231.
[43] Powell, Daniel. "Tool support for verification-based software inspection." Software Engineering Conference, 2004. Proceedings. 2004 Australian. IEEE, 2004.
[44] Hussain, Fida, and Muhammad Saeed Shehzad. "" Robust and Flexible Software Inspection model" for Software Re-Engineering Process: Abstraction phase." 14th Asia-Pacific Software Engineering Conference (APSEC'07). 2007.
[45] Hashemitaba, Navid, and Siew Hock Ow. "Generative inspection: an intelligent model to detect and remove software defects." Intelligent Systems, Modeling and Simulation (ISMS), 2012 Third International Conference on. IEEE, 2012.
[46] Christenson, Dennis A., and Steel T. Huang. "A code inspection model for software quality management and prediction." Global Telecommunications Conference, 1988, and Exhibition.'Communications for the Information Age.'Conference Record, GLOBECOM’88, IEEE. IEEE, 1988.
[47] Biffl, Stefan, Michael Halling, and Monika Kohle. "Investigating the effect of a second software inspection cycle. Cost-benefit data from a large-scale experiment on re-inspection of a software requirements document." Quality Software, 2000. Proceedings. First Asia-Pacific Conference on. IEEE, 2000.

Authors’ Profiles

Asad Masood Qazi is the Research Student of University Institute of Information Technology – PMAS – Arid Agriculture University Rawalpindi. He is doing his MS degree in Software Engineering. He is also an Author of 02 research articles as well, related to Software Architecture and Software Process Improvement.

Author also has an experience to work in Software Development & Consultancy organization as Business Intelligence Consultant to develop the business intelligence / Data warehouse related software projects of different clients (National as well as International).

The major area of interest in Research is Software Engineering Principles, Processes, Software Quality Assurance, Knowledge Management, Formal methods of software engineering and Software Architecture.

Ms. Sidra Shahzadi is a full time Research Student at University Institute of Information Technology – PMAS – Arid Agriculture University Rawalpindi. She has completed her course work and involves in different research oriented activities conducted by Institute.

Author has interest in different research domains of Software Engineering related to Requirements Engineering, Formal Methods in software engineering, Role of Ontologies and Modelling techniques in Software engineering domain.

Dr. Mamoona Humayun is Assistant Professor at University Institute of Information Technology – PMAS – Arid Agriculture University Rawalpindi. She has more than 05 years of experience in Teaching, Research and other academic activities. She has completed her PHD Degree from Harbin Institute of Technology Harbin, China in the field of Software Engineering.

Dr. Mamoona Humayun is an author of more than 10 research articles and her major areas of interest are related to Global Software Development, Software Requirements Engineering, Knowledge Management, Software Testing and Web Applications Security.