Framework for the Development of Automated Inspection Tools

Muhammad Shahid Iqbal[†], Muhammad Sadiq[††], Abdul Rehman[†††] and Tamoor Khan[††††]

[†]School of Computer Science, Anhui University, Hefei, China
[††]Faculty of Computing (RIU) Islamabad, Pakistan
[†††]College of Economics & Management, Anhui Agricultural University Hefei, China
[††††]Master Student, Leads University Lahore, Pakistan

Summary

In early stage of software development inspection is one of the best method for identifies defects and removing defects. Automated inspections are done with some automated inspection tools. A number of automated inspection tools have been developed to support software inspection. Meanwhile existing automated inspection tools have implemented few set of parameters and software development environment require many parameters for inspection process. In this paper, we describe some common refer tool on the basis of literature also identified parameter and sub parameter of these tools. Then we conduct an industry survey meanwhile we combined both surveys. Then we proposed a framework for the development of automated inspection tools which set of parameters are implemented by automated inspection tools.

Key words:
Automated Tools, Parameters, Inspection Tools, Framework, Anomaly classification & Interoperability.

1. Introduction

Inspection is simple method for defect identified form artefacts which can be done in any phase of software developments [94]. F. Macdonald, J. Milleret et al authors compared automated tools which support inspection process and conclude that no single tool available fills all the identified needs of inspection. Furthermore author suggested features like Document support, annotation support, checklists, enforcement, and distributed meeting support, polls and metrics collection as part of inspection tool [14]. M. Halling, P. Grünbacher et al, authors compare existing inspection tools with groupware support system technology and then provides a flexible and powerful set of tools to support the entire inspection process [72]. Filippo Lanubile, Teresa Mallardo et al authors discuss importance of communication. There are two type of communication synchronous and asynchronous discussion whereas he shows that asynchronous meeting is more effective then synchronous [6]. F. Bomarius and H. Iida et al, among all the parameter of static testing author emphasize on flexibility and integration. Two additional features are being proposed for inspection tools [21]. A. De Lucia F. Fasano et al, Authors' made a comparison of static testing tool after the comparison author purpose discipline and flexibility as additional parameters of static testing tool[1].

2. Existing Automated Tools

Most of automated inspection tools implemented variety of features, documentation, meeting, communication, anomaly identification and inspection checklist in the field of automated testing. Some tools focus on one or two parameters and some of them focus on many parameters which we mention in detail as following [1], [20].

2.1 ASSIST

ASSIST used custom designed language which is known as "Inspection Process Definition Language" ASSIST allows to any inspection process [13].

Document
ASSIST allowing all types of document text, code and graphic. It added associated browsers when it required and also support several browsers. Meanwhile allows using such type of features like annotation etc. [1], [2], [13], [20], [56], [72].

Individual Preparation
ASSIST have private list, every inspector studying the product and adding errors/defects on their private list [68], [38].

Meeting support
It supports meeting synchronous and asynchronous [1], [2], [20].

Data Collection
ASSIST supports data collection automatically [68], [38].

Manuscript received January 5, 2016
Manuscript revised January 20, 2016

Checklists
ASSIST provides a checklist bowser which implementing active check list to record answer of inspection [1],
[2], [13], [20], [72].

Cross Referencing System
ASSIST provides cross referencing system to show the same word appearing in different documents. It link to related part of documents together. It could provide a means of navigating within documents. Automatic cross-referencing allows inspectors to easily move within and between documents to find specific features. [1][2][13][16][55].

Defect Classification
It provides the feature to automatically classify the defects [1] [20].

Defect Detection
ASSIST allows automatic defect detection [1] [20][38][68][72].

Voting Facility
It allows its user to vote for certain class of defect [20].

E-mail Notification
It allows sending e-mail notification to review team member [1].

Decision Support
ASSIST allows decision support facility [1].

2.2 ICICLE

In this automated inspection tools authors attempts to replace manual inspection. It is an automated intelligent inspection assistant developed to support the inspection of C and C++ code [1][20].

Defect Classification
ICICLE classified defect automatically [1] [20].

Cross Referencing
It provides cross reference such as variable and function. When click on a variable it give an option to move on declaration point. This facility CSRS provide for many source files [1] [20].

Data Collection
Under inspection product ICICLE generates a list of all accepted defects. Also generated summery of defects and summery have information defect type class and severity. A summary of the defects by type, class and severity is also generated which contain such type of information total time spent in perpetration and meeting [1][20].

Discussion Support
It supports discussion. The discussion in signal room it is very simple way and tool allow each inspector to propose comments, also record the outcome [1][20]

Document support
ICICLE support only source code and text document [1][20]

2.3. CSRS

CSRS support formal technical asynchronous review method (FTArm). FTArm is a technique for inspection [19].

Document Handling
Records are put in to database and arrange like hubs. Hubs compares the source code function, variables also changes in records. CSRS only supports text type documents [1], [20].

Decision Support
It supports decision through available polls. It supports asynchronies discussion [1].

Automatic E-mail Notifications.
It provides E-mail facility to all reviewer automatically send message to all reviewers when new node are created [1].

Check list Support
It supports on line checklist and chick list focus on main issue also their types meanwhile assist to the reviewer [2].

Voting Facility
It allow to the inspectors vote about bug meanwhile discuss level of bug like major, minor or critical [20].

2.4 Scrutiny

Scrutiny is use for distributed and collaborative inspection meanwhile artifacts review. Scrutiny process is similar to the Fagan's process. It supports face to face inspection. It maximizes team inspections and individual [1], [2].

Data Collection
Scrutiny has capability to gather comprehensive matric. It has ability to gather defect metrics, as well as fine-grained metrics on the amount of time spent by each inspector reviewing each node, the time spent in inspection and the coverage of the document achieved by each inspector[1][20] .

Voting Facility
Scrutiny has facility of voting and allows to inspection team to vote about errors [20].

Document
Scrutiny supports only source code documents [1] [20].

Automatic message facility
Scrutiny sends simple messages to meeting participants meanwhile compose own message. Message can be send to single participant or whole inspection team [20].

2.5 Collaborative Software Inspection (CSI)

It supports online inspection meanwhile it support face to face meeting and distributed meeting [20]. CSI supports all types of documents e.g., text, code and graphics [20].

Metric Collection
Meanwhile it gives additional history in history log it contains information like discover fault and claiming faults found [1].

Defect Classification
Also classifications faults. [1], [20]

Data collection
It runs down judgment to record the meeting data. [1], [20].

Meeting Support
CSI support distributed meeting. It allows an inspection meeting to be carried out with team members from different locations [20].

2.6 Asynchronous Inspector of Software Artifacts (AISA)

ASIA structured allows inspection of graphical object meanwhile it supports distributed environment. ASIA supports three stages of inspection process defect collection, defect correlation and inspection meeting [1], [20].

Defect Classification
AISA classified the defect and contain information about defects for this purpose used HTML template [1], [20].

Voting Facility
It give right to Participants can vote to accept or reject defects [20].

Document Support
ASIA supports three types of document text, source code and graphics just like entity relationship diagrams and data flow diagram [68].

Distributed meeting supports
It support distributed meeting [1], [20].

Flexibility
Flexibility of the inspection process and inspection tools means independence of time and place. Flexibility has two most significant features for implementing the next generation of inspection tools. WWW technology was chosen due to its popularity, familiarity and flexibility [71], [10], [68], [69].

Interoperability
Interoperability of the processes and tools, to enable convenient everyday use of the method and improves the effectiveness of inspections. The most important enhancement that inspection tools need is interoperability [10].

Defect Correlation
The producer integrates individual defect lists into a single master list [38].

2.7 Collaborative Asynchronous Inspection of Software (CAIS)

It support distributed environment for software inspection also support asynchronous discussions. CAIS create defects list and organize meeting for contributions (votes and comments) to discussion [1].

Individual Preparation
CAIS and CSI both are identical individual preparation. It supports asynchronous meeting meanwhile it gives voting mechanism [20].

Data Collection
It use history log for gather information about software review. Also maintain the record of reviewer remarks and defect classification [1].

Defect Collection
It provides the facility to automatically collect the defect [20].

Voting Facility
It allow to the inspector give vote about bugs [20].

E-mail Notification
Through E-mail; it notifies the each participant when new discussion has taken place [1], [20].

Decision support
CAIS have features to take decision about bug it is critical, major and minor [1]

Document Support
CAIS support three types of documents like source code, text and graphic [1], [20].

2.8 Inspecting Software in Phases to Ensure Quality (InspeQ)

Knight and Myers developed InspeQ tool set which support inspection technique. They also propose technique for artifacts which examine artifacts in series according to the inspection phases. This technique is implemented in InspeQ [1], [20].

Document Handling
InspeQ supports three types of documents source code, text and graphic [1].

Checklist Support
It display checklist of current inspected software meanwhile in checklist each inspector can mentation completion of each check [1], [20].

Source Code
It supports inspection of source code which is in C languages [1].

2.9 Inspect A

It supports asynchronous inspection and start from individual inspection, where inspectors generate their initial list of comments. Meanwhile this list is share with each inspector also allowing discussing validity of each comment, at end phase prepare a master list and send each inspector [1], [20].

Checklist Support
Inspect A used inspection checklist for the review purpose [1].

E-mail Notification
E-mail generation facility is available to notify the inspector about inspection completion [1], [2].

Document Handling
Inspect A supports just plain content documents. It also permits a rundown from claiming Defects on be entered. Every deformity might incorporate the result quick which will be incorrect, a depiction. Of the defect, a population (Missing, off alternately Extra) Furthermore a seriousness (Major alternately Minor). The Defects would not connect of the position in the archive the place they happen [1].

2.10 Hyper Code

It is a web base tool however preparation and collection are performed at the same time. Inspection team member are inform through email when inspection will be started [1], [20].

Document Handling
It supports three types of documents text, code and graphical [1], [20].

Email- notification
E-mail notification facility to notify the team members when inspection will be started [1].

2.11 Automated Static Analysis Tools

Automatic static analyses tools analyze source code searching for violations for bug designs that might result in faulty conduct technique. ASA uses control flow analysis, information stream analysis, interface analysis, majority of the data flow analysis and furthermore way examination from reach product code. There will be a range of programmer errors which can be automatically detected by ASA [73].

Identifying Defects
Static analysis tools have been used for identifying defects in software systems. There is a range of programm errors which can be automatically detected by ASA [73], [74], [75], [76].

Efficiency
The developers utilization ASA on check code compliance should standards or alternately should assess the inside personal satisfaction and kill conceivable wellsprings of slip Also wastefulness of the formed framework [74], [76].

Error Free
ASA can find the error from software and eliminated the error of any type [76].

Performance
It has potential to reduce code volume which have identified bugs via unused code [75], [76].

Correctness
It identifies syntax error as well as interface error [75], [76].

Quality
It focuses on coding standards and also enforce on architectural [76].

2.12 Finding Bugs Tool

Finding Bugs is an open source static analysis tool that analyzes Java class files looking for programming error. FB has a plugin architecture, in which detectors can be defined, each of which may report many different bug patterns. Rather than use a pattern language for describing bugs. An FB detector is simply written in Java, which use many methods.

Defect Detection
Bugs are detected by FB and it finds all possible bugs it also provides relation between bugs [79].

Performance
Find Bugs looks good to improve code quality because it detected not only bugs, but also bad programming practices [79].

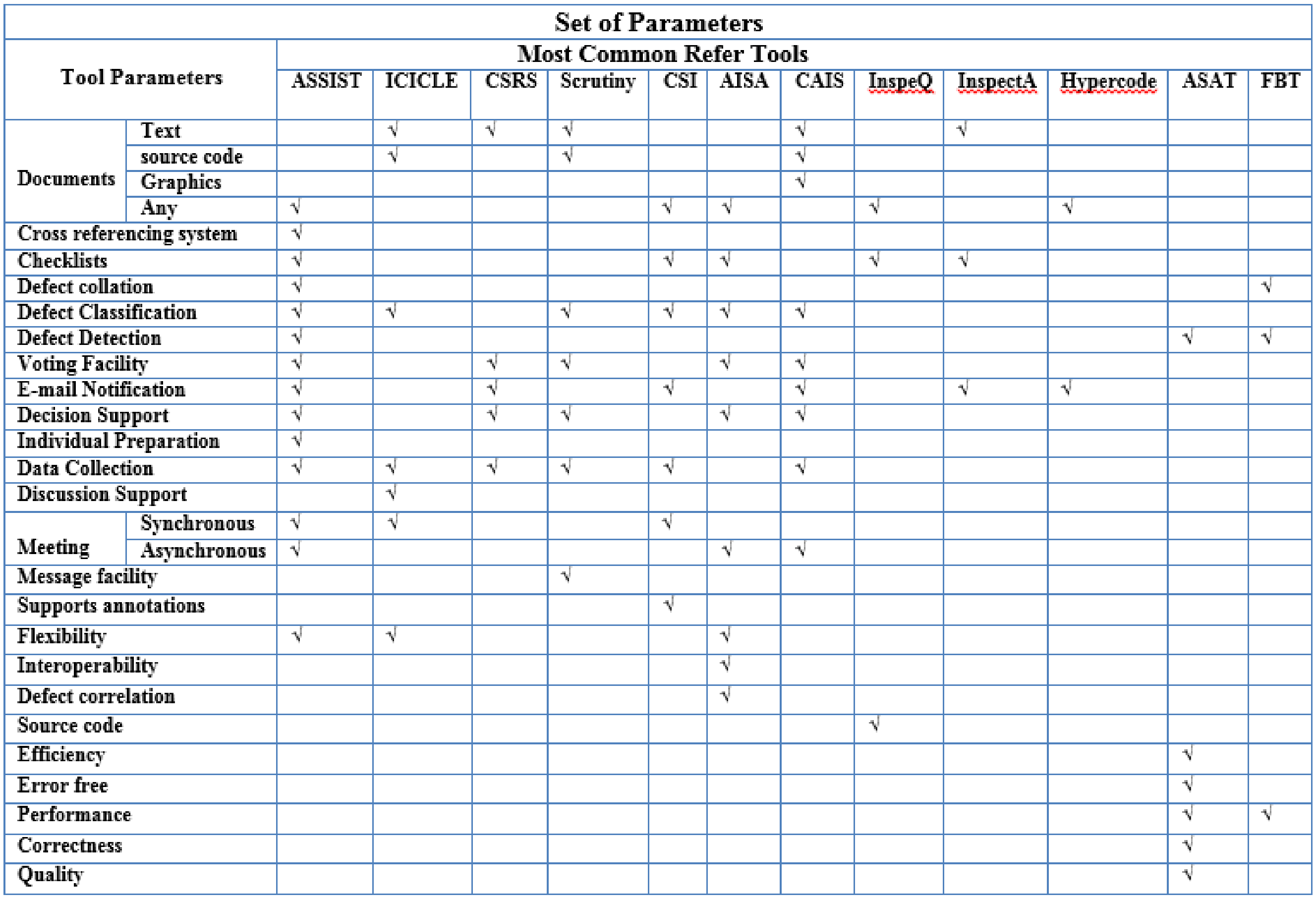
Defect Collection
By review FB detect error it type. It is use FB tool to find subset of defects [80].

Data Collection
In order to collect the data for the evaluation of degree of static [81], [82].

3. Set of Parameters Identified from literature

These are the twelve common refer tools which are identified from literature and minimum set of parameters for the inspection tools. ASSIST support all type of document and also implemented so many features just like data collection, defect collection, voting facility, e _mail etc. ICICLE has only two types of documents. It also classified the errors furthermore it support data collection and discussion support, face to face meeting finally it support flexibility. CSRS only support text type document in additionally it supports voting facility, data collection and e-mail notification. Scrutiny implements two types of document text and source code. It also classified the defects as well as data collection, voting facility and discussion support. CSI support all type of document finally it implement so May parameters like checklist, defect classification and email. AISA implemented all type of document and it support inspection checklist data collection etc. All tool are implemented few features. Document is only parameters which are implemented by ten tools. Defect classification implemented by eight tools. Email and data collection is implemented by six tools. The rest of parameters are less the five tools are implemented. Hence conclude that all of tools implemented few parameters and above table give road map for the development of inspection tools which have minimum set of parameters are shown in Table 1.

Table 4.1 √ = Tool implement the features



Set of Parameters													
Tool Parameters		Most Common Refer Tools											
		ASSIST	ICICLE	CSRS	Scrutiny	CSI	AISA	CAIS	InspeQ	InspectA	Hypercode	ASAT	FBT
Documents	Text		√	√	√			√		√			
	source code		√		√			√					
	Graphics							√					
	Any	√				√	√		√		√		
Cross referencing system		√											
Checklists		√				√	√		√	√			
Defect collation		√											√
Defect Classification		√	√		√	√	√	√					
Defect Detection		√										√	√
Voting Facility		√		√	√		√	√					
E-mail Notification		√		√		√		√		√	√		
Decision Support		√		√	√		√	√					
Individual Preparation		√											
Data Collection		√	√	√	√	√		√					
Discussion Support			√										
Meeting	Synchronous	√	√			√							
	Asynchronous	√					√	√					
Message facility					√								
Supports annotations						√							
Flexibility		√	√				√						
Interoperability							√						
Defect correlation							√						
Source code									√				
Efficiency												√	
Error free												√	
Performance												√	√
Correctness												√	
Quality												√	

4. Industry Survey

We conducted an industry survey internationally through questioner. First we will classify the parameters according to the inspection process and then classify these parameter in such way i.e. inspection input, inspection objective, inspection planning, communication with inspection team, data collection and inspection performance measurement.

Inspection Input

From literature survey the following parameters are includes “Specification document e.g. requirement / design and Inspection check list” these two parameters are identified from literature and we are includes in the first phase of inspection. We post a question from industry which types of input are used in your industry. Most of software houses are sport the document are must be included in automated inspection tool. Most of software industry response that the documents are used in their organization and few are support checklist. Most of software industry support documentation this parameter must be included in automated inspection tool and only twelve percentages software industry support inspection check list features are shown in Figure 1.

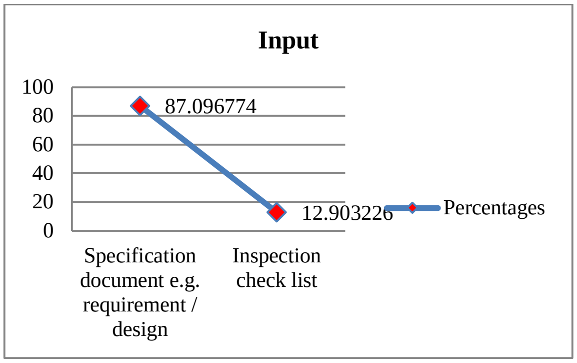


Figure 1

Inspection Objective

Inspection has following four parameters which are identified form literature accuracy, flexibility, completeness and interoperability. Most of the industries suggest that accuracy and completeness are included in inspection automated tools.

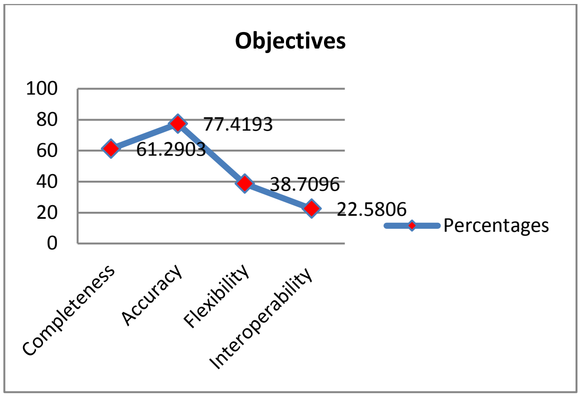


Figure 2

Planning

The next phase is planning and planning includes two parameters resource planning and task allocation. In industry response seventy percent support resource planning is included in automated inspection tool and seventy seven percent industry support task allocation. It means both parameters are must be included for next generation of automated inspection tools.

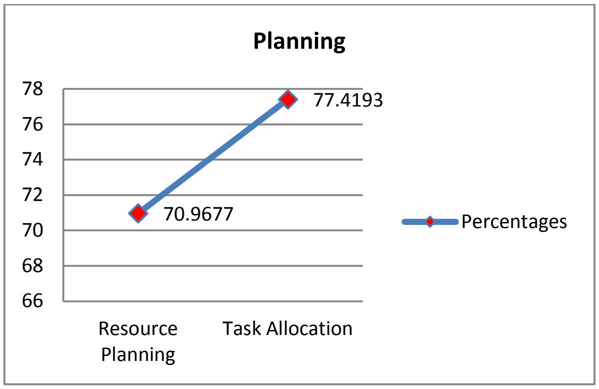


Figure 3

Communication

Inspection team communication is one of the important phase of inspection process in this process following parameters are includes F2F meeting, asynchronies meeting, email notification, voting facility and discussion. In this industry survey 64% F2F, 32% distributed, 71% percent email, 32% percent voting facility and 9 ½% software industries support these parameters.

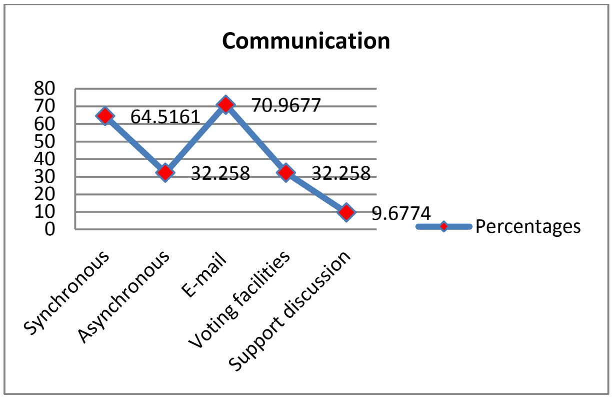


Figure 4

Data collection

Data collection phase include following parameters which are evaluated from industry's anomaly record, errors classification, anomaly ranking and anomaly correlation. Industry's response fifty eight percent anomaly record, seventy seven percent error classification, twenty five percent both anomaly ranking and anomaly correlation.

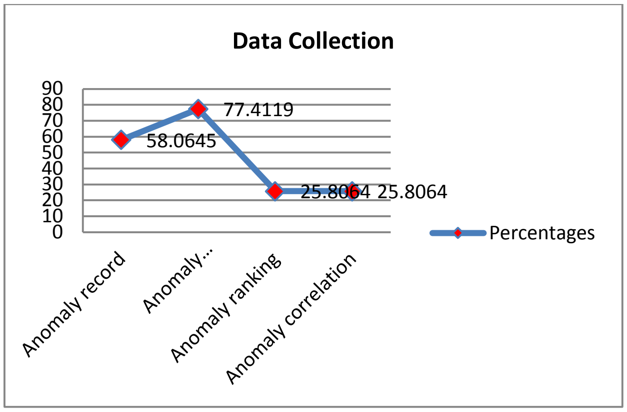


Figure 5

Inspection measurement

In this process following parameters are included completeness, accuracy, flexibility and interoperability through industry survey following response from industry's 71% support completeness, 74% accuracy, 25% flexibility and 35% interoperability are included in inspection tools.

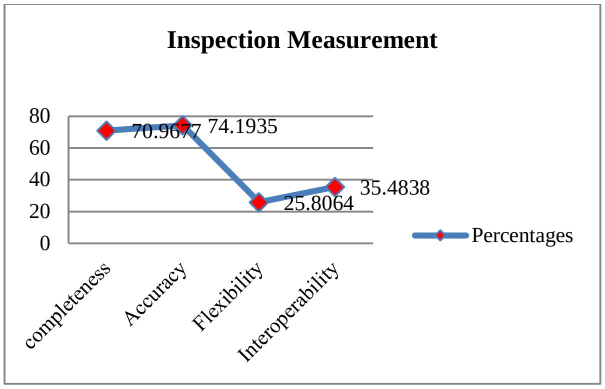


Figure 6

5. Framework for Future Development

On the bases of industry survey we proposed the following parameters are must be included in automated inspection tools for future generation. Following parameters are the requirement of software industry's "Specification document e.g. requirement / design, Completeness, Accuracy, Resource Planning, Task Allocation, E-mail notification/ Massage facilities, Synchronous/ Face to face, Anomaly record, Anomaly classification/error classification, completeness, Accuracy". These parameters should be included in future generation of automated inspection tools. These parameters are selected one the base of percentage each parameter response is above 50 percent.

6. Conclusions

In this research, we have proposed framework for the development of automated inspection tools. This framework achieve what set of parameters are required to develop automated inspection tools. All tools have implemented few parameters not completed set of parameters Also parameters are classified on the bases of inspection process. Set of parameters are done through literature survey, identified common refer automated inspection tools and their parameters. Meanwhile we conducted industry survey then done analysis on the bases of both survey. We proposed a minimum set of parameters which must be in in any automated inspection tool. In future work in same way can be describes the gaps of other testing tools like performance testing, functional testing and test case management etc.

References

[1] De Lucia F. Fasano G. and Scanniello G. Tortora. Evaluating distributed inspection through controlled experiments. IET Software, Vol. 3, Issue. 5, pp. 381–394, 2009

[2] De Lucia F. Fasano G. and Scanniello G. Tortora Assessing the Effectiveness of a Distributed Method for Code Inspection: A Controlled Experiment, International Conference on Global Software Engineering (ICGSE) , IEEE computer socitety, 2007

[3] Porter, L.G. Votta and V.R. Basili, Comparing detection methods for software requirements inspections: A replicated experiment, IEEE Transactions on Software Engineering 21(6) , 563–575, 1995

[4] Porter and P. Johnson, Assessing software review meetings: Results of a comparative analysis of two experimental studies, IEEE Transactions on Software Engineering 23(3) (1997) 129–145.

[5] Bell Communications Research. ICICLE User's Guide, January 1993.

[6] Bull HNInformation Systems, Inc., U.S. Applied Research Laboratory. Scrutiny User's Guide, May 1994.

[7] D. Tjahjono. Comparing the cost effectiveness of group synchronous review method and individual asynchronous review method using CSRS: Results of pilot study. Technical Report ICS-TR-95-07, University of Hawaii, January 1995

[8] D.L. Parnas and D.M. Weiss, Active design reviews: Principles and practices, in: Proceedings of the 8th International Conference on Software Engineering (August 1985) pp. 132–136, 1985

[9] D. Tjahjono, Exploring the Effectiveness of Formal Technical Review Factors with CSRS, a Collaborative Software Review System, PhD thesis, Department of Information and Computer Sciences, University of Hawaii, June 1996.

[10] D. Mishra and A. Mishra. A Software Inspection Process for Globally Distributed Teams Department of Computer Engineering, Atilim University, Springer-Verlag Berlin Heidelberg, pp. 289–296, 2010

[11] D. Hovemeyer and W. Pugh, Finding Bugs is Easy, In SIGPLAN Notices, vol. 39, pp. 192-206, 2004

[12] E. A. Meyers and J. C. Knight. An improved software inspection technique and an empirical evaluation of its effectiveness. Technical Report TR-92-15, Department of Computer Science, University of Virginia, May 1992.

[13] F. Macdonald, J. Miller "ASSIST—a tool to support software inspection" Empirical Foundations of Computer Science (EFoCS), Department of ComputerScience, University of Strathclyde, Glasgow, G1 1XH, UK, Information and Software Technology volume 41 pages 1045–1057, 1999

[14] F. Macdonald, J. Miller, A. Brooks, M. Roper, M. Wood, A review of tool support for software inspection, Proceedings of the Seventh International Workshop on Computer Aided Software Engineering, pp. 340–349, 1995

[15] F. Macdonald, J. Miller, A. Brooks, M. Roper, M. Wood, Automating the software inspection process, Automated Software Engineering: An International Journal Volume 3, No.3/4 pp.193–218. 1996

[16] J. Miller *, F. Macdonald, An Empirical Incremental Approach To Tool Evaluation And Improvement, The Journal of Systems and Software 51 pages 19-35, 2000

[17] F. Macdonald and J. Miller. A software inspection process definition language and prototype support tool. Software Testing, Verification and Reliability, volume 7, No.2: pp. 99-128, June 1997.

[18] F. Macdonald, J. Miller, A comparison of tool-based and paper-based software inspection, Empirical Software Engineering: An International Journal 3 (3) (1998) 233–253

[19] F. Macdonald, J. Miller, A. Brooks, M. Roper and M.Wood, A review of tool support for software inspection, in: Proceedings of the 7th International Workshop on Computer Aided Software Engineering (July 1995) pp. 340–349, 1995

[20] F. Lanubile, T. Mallardo and F. Calefato, Tool Support for Geographically Dispersed Inspection Teams, Softw. Process Improve. Pract. volume 8: pages 217–231 , 2003

[21] F. Bomarius and H. Iida. Introducing the Next Generation of Software Inspection Tools Henrik Hedberg, Springer-Verlag Berlin Heidelberg, pp. 234–247, 2004.

[22] Gintell J.W., Arnold J., Houde M., Kruszelnicki J., Mckenney R., Memmi G. Scrutiny: a collaborative inspection and review system. Proc. Fourth European Conf. on Software Engineering, pp. 344–360, 1993

[23] Harjumaa, L. and Tervonen, I. (1998). A WWW-based tool for software inspection. Proceeding of the Thirty-first Hawaii International Conference on System Sciences (HICSS'98), Vol. 3. Los Alamitos, CA: IEEE Press, 379-88, 1998

[24] Humphrey W.S.: Managing the software process SEI series in software engineering, Addison-Wesley Longman Publishing, Boston, MA, USA, 1989

[25] J.W. Gintell, J. Arnold,M. Houde, J. Kruszelnicki, R.McKenney and G.Memmi, Scrutiny: A Collaborative Inspection and Review System, in Proc. 4th European SoftwareEngineering Conference, Garwisch-Partenkirchen,Germany, September 1993.

[26] J. Reidl, V. Mashayekhi, J. Schnepf, M. Claypool, D. Frankowski, SuiteSound- A System for Distributed Collaborative Multimedia, IEEE Transactions on Knowledge and Data Engineering, pp. 600-610, Vol. 5, No. 4, 1993.

[27] Johnson, P. An instrumented approach to improving software quality through formal technical review. Proceeding of 16th International Conference Software Engineering (ICSE'94). Los Alamitos, CA: IEEE CS Press, 113-22, 1994

[28] 15. Johnson, P., & Tjahjono, D. Assessing software review meetings: A controlled experiment study using CSRS. Proceeding of the 19th International Conference on Software Engineering (ICSE'97). Los Angles, CA: ACM Press, 118-27, 1997

[29] J.W. Gintell, J. Arnold, M. Houde, J. Kruszelnicki, R. McKenney, G. Memmi, Scrutiny: a collaborative inspection and review system, Proceedings of the Fourth European Software Engineering Conference, September 1993.

[30] [k] J. Miller, F. Macdonald, An empirical incremental approach to tool evaluation and improvement. Journal of Systems and Software (in press).

[31] Knight J.C., Meyers E.A.: An improved inspection technique, ACM, 36, (11), pp. 51–61,1993

[32] Knight J.C., Meyers E.A. Phased inspections and their implementation, Software. Engg. Notes, 16, (3), pp. 29–35,1991

[33] L. R. Brothers, V. Sembugamoorthy, and A. E. Irgon. Knowledge-based code inspection with ICICLE. In Innovative Applications of Artificial Intelligence 4: Proceedings of IAAI-92, 1992.

[34] L. Brothers, V. Sembugamoorthy and M. Muller, "ICICLE: Groupware for Code Inspections," in Proc. 1990ACMConference on Computer SupportedCooperative Work, pp. 169-181, October 1990.

[35] L.R. Brothers, V. Sembugamoorthy, M. Muller, ICICLE: groupware for code inspections, Proceedings of the ACM Conference on Computer Supported Cooperative Work, pp. 169–181, 1990

[36] Macdonald and J. Miller : A Comparison of Tool-Based and Paper-Based Software Inspection F _ISERN-98-17 April 1997

[37] Mashayekhi V., Drake J.M., Tsai W.T., Reidl J.: Distributed, collaborative software inspection, IEEE Software, 10,(5), pp. 66–75, 1993

[38] MacDonald, F. Computer Supported Software Inspection. PhD Thesis. Department of Computer Science, University of Strathclyde, UK., 1998

[39] MacDonald, F., & Miller, J. Automated generic support for software inspection. Proceedings of the 10th International Quality Week., 1997

[40] Macdonald, F., & Miller, J. A comparison of tool-based and paper-based software inspection. Empirical Software Engineering 3(3), 233-53.,1998

[41] MacDonald, F., & Miller, J. Automated generic support for software inspection. Proceedings of the 10th International Quality Week, 1997

[42] P. Johnson, Supporting Technology Transfer of Formal Technical Review Through a Computer Supported Collaborative Review System, in Proc. 16th International Conference on Software Quality, Reston, VA, October 1994.

[43] Stein, M., Riedl, J., Harner, S.J., & Mashayekhi, V. (1997). A case study of distributed, asynchronous software inspection. Proceeding of the 19th International Conference on Software Engineering (ICSE'97). Los Angeles, CA: ACM Press, 107-17, 1997

[44] T. Gilb and D. Graham, Software Inspection, Addison-Wesley, 1993.

[45] Tervonen, I. (1996). Support for quality-based design and inspection. IEEE Software 13(1), 44-54., 1996

[46] V. Mashayekhi, J. M. Drake, W. T. Tsai and J. Reidl, Distributed, Collaborative Software Inspection, IEEE Software, Vol. 10, No. 5, pp. 66-75, September 1993.

[47] V. Mashayekhi, C. Feulner and J. Reidl, CAIS: Collaborative Asynchronous Inspection of Software, in Proc. 2nd ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 21-34, 1994.

[48] V. Sembugamoorthy and L. R. Brothers. ICICLE: Intelligent Code Inspection in a C

[49] Language Environment. In Proceedings of the 14th Annual Computer Software and Applications Conference, pages 146–154, October 1990.

[50] Scrutiny User's Guide: A Distributed System for Collaborative Inspection and Review of Work Products,

USARL/94-1, Bull HN Information Systems Inc., Applied Research Laboratory, June 1994

[51] Murphy P., Miller J.: ‘A process for asynchronous software inspection’. Proc. Eighth Int. Workshop on Software Technology and Engineering Practice, London, UK, pp. 96–104, 1997

[52] Johnson P.M.: ‘An instrumented approach to improving software quality through formal technical review’. Proc. 16th Int. Conf. on Software Engineering, Sorrento, Italy, pp. 113–122, 1994

[53] Yamashita T. Evaluation of Jupiter: a lightweight codereview framework. M.S. thesis, University of Hawaii, Honolulu, Hawaii, Number CSDL-06-09, December, 2006, available at http://csdl.ics.hawaii.edu/techreports/06-09/

[54] Perpich J.M., Perry D.E., Porter A.A., Votta L.G., Wade M.W.: Anywhere, anytime code inspections: using the web to remove inspection bottlenecks in large-scale software development’. Proc. 19th Int. Conf. on Software Engineering, Boston, Massachusetts, USA, pp. 14–21

[55] J. Miller , F. Macdonald An empirical incremental approach to tool evaluation and improvement The Journal of Systems and Software 51, 19±35 Elsevier, 2000

[56] MacDonald, F., Miller, J., Ferguson, J., “ASSISTing Management Decisions in the Software Inspection Process”, Journal of Information Technology and Management, Vol. 3, pp. 67-83, 2002.

[57] J.W. Gintell, J. Arnold, M. Houde, J. Kruszelnicki, R. McKenney and G. Memmi, Scrutiny: A collaborative inspection and review system, in: Proceedings of the 4th European Software Engineering Conference, 1993.

[58] J. Miller, M. Roper and M. Wood, Further experiences with scenarios and checklists, Empirical Software Engineering (3) 37–64, 1998

[59] L.G. Votta, Does every inspection need a meeting? in: Proceedings of the First ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 107–114, 1993

[60] MacDonald F., Miller J., A Comparison of Computer Support Systems for Software Inspection, Automated Software Engineering 6, 291-313, 1999.

[61] Harjumaa L. and Tervonen I., A WWW-based tool for software inspection, Proc. of HICSS-98, vol. 3, 1998.

[62] M. Halling, S. Biffl P. Grünbacher. An Experiment Family to Investigate the Defect Detection Effect of Tool-Support for Requirements Inspection Proceedings of the Ninth International Software Metrics Symposium (METRICS’03) 1530-1435/03, IEEE computer society, 2003

[63] L. Harjumaa, oululfi, H. Hedberg I. Tewonen. A Path to Virtual Software Inspection Department of Information Processing Science, University of Oulu, P.O.Box 3000, FIN-90014 OULUN YLIOPISTO 3, IEE

[64] Stein, M., Riedl, J., Harner, S.J., & Mashayekhi, V. A case study of distributed, asynchronous software inspection. Proceeding of the 19th International Conference on Software Engineering (ICSE’97). Los Angeles, CA: ACM Press, 107-17, 199

[65] V. Mashayekhi, C. Feulner, and J. Reidl. CAIS: Collaborative Asynchronous Inspection of Software. In Proceedings of the Second ACM SIGSOFT Symposium on the Foundations of Software Engineering, December 1994.

[66] J. C. Knight and E. A. Meyers. Phased inspections and their implementation. Software Engineering Notes, 16(3):29–35, July 1991.

[67] J. C. Knight and E. A.Meyers. An improved inspection technique. Communications of the ACM, 36(11):51–61, November 199

[68] Demirhan and Taylan," A study on developing a software inspection methodology". In the journal Software Engineering Journal ,June 2006, 92 pages

[69] Bordin Sapsomboon," Software Inspection and Computer Support ". In the Journal Automated Software Engineering Volume 6 Issue 3, july 1999, Pages 291-313

[70] Sami Kollanus and Jussi Koskinen," Survey of Software Inspection Research". In the journal Open Software Engineering Journal, Volume: 3, Issue: 1, 2009 Science Pages: 15-34

[71] Micheal Spring and Bordin Sapsomboon, " Shared defect detection: the effects of annotations in asynchronous software inspection". In the Proceeding doctoral dissertation shared defect detection: the effects of annotations in asynchronous software inspection, Pages 206, 2000.

[72] James Miller, John D. Ferguson and Murphy, "Groupware Support for Software Requirements Inspection". In the Proceeding 17th annual international conference on Computer Documentation, SIGDOC 1999, 1999 Pages:185-192

[73] Jiang,Nachiappan Nagappan ,Laurie William ,A.vouk and J.Hudepol," On the Value of Static Analysis for Fault Detection in Software". In the journal IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 32, NO. 4, APRIL 2006.

[74] M. Vetro', A.; Torchiano, M; Morisio (2011): "Quantitative Assessment of the Impact of Automatic Static Analysis Issues on Time Efficiency". In Quantitative Information 2011.

[75] Nathaniel, David, J.David, John and William Pugh," Experiences Using Static Analysis to Find Bugs". In the Journal IEEE Volume 25 Issue 5,September 2008 Pages 22-29

[76] Nachiappan,Laurie,J.Hudepohl,Will Snipes and Malden Vouk," Preliminary Results On Using Static Analysis Tools For Software Inspection". In the Proceeding ISSRE'04 Proceedings of the 15th International Symposium on Software Reliability Engineering Pages 429-439.

[77] Dr Paul Anderson "The Use and Limitations of Static-Analysis Tools to Improve Software Quality". In the Journal of Defense Software Engineering ,June 2008

[78] Nathaniel Ayewah, William Pugh, J. David Morgenthaler, John Penix, YuQian Zhou: Evaluating Static Analysis Defect Warnings On Production Software. In Proceeding of PASTE '07 Proceedings of the 7th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering. New York, NY, USA ©2007.

[79] HunJae Lee, SeongYong Lim, SeJoon Oh and Duri Kim," Mini-Project: Tool or Analysis Practicum", April 7,2009.

[80] Stefan Wagner, Jan Jurjens,Claudia Koller and Peter Trischberger "Comparing bug finding tools with reviews and tests". In the Proceeding TestCom'05 Proceedings of the 17th IFIP TC6/Wg6.1 International conference of testing of communication system Pages 40-55, 2005.

[81] Cesar Couto and Christopher Silva "Static correspondence and correlation between field defects and warnings reported by a bug finding tool" .In the journal Software quality journal LLC,2011.

[82] Antonio ,Maurizio and Marco "An Empirical Validation of FindBugs Issues Related to Defects". In the proceeding Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on Pages 11-12, April 2011

[83] Par Emanuel son and Ulf Nilsson "A Comparative Study on Software Vulnerability Static Analysis Techniques and Tools". In the Journal Electronic Notes in Theoretical Computer Science Volume 217 Pages 5-21 ,July 2008.

[84] http://istqbexamcertification.com/what-is-verification-in-software-testing-or-what-is-software-verification/

[85] http://swtestingbasics.blogspot.com/2009/03/static-testing-or-reviews.html

[86] R. L. Glass, R. Collard, A. Bertolino, J. Bach, and C. Kaner, “Software testing and industry needs,” IEEE Software, vol. 23, no. 4, pp. 55–57, 2006.

[87] Rafi, D.M.; Moses, K.R.K.; Petersen, K.; Mantyla, M.V., "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey," Automation of Software Test (AST), 2012 7th International Workshop on , vol., no., pp.36,42, 2-3 June 2012

[88] http://www.qualitysystems.com/support/pages/10-management-review

[89] Misha Zitser, Richard Lippmann, Tim Leek, Testing Static Analysis Tools using Exploitable Buffer Overflows from Open Source Code, SIGSOFT’04/FSE12, Oct. 31–Nov. 6, 2004, Newport Beach, CA, USA.Copyright 2004 ACM 1581138555/ 04/0010 ...$5.00.

[90] Fadi Wedyan, Dalal Alrmuny, and James M. Bieman, The Effectiveness of Automated Static Analysis Tools for Fault Detection and Refactoring Prediction

[91] Sarah Heckman and Laurie Williams, A Model Building Process for Identifying Actionable Static Analysis Alerts

[92] http://searchsoftwarequality.techtarget.com/definition/automated-software-testing

[93] FREEDMAN D.P., WEINBERG G.M.: ‘Handbook of walkthroughs,inspections, and technical reviews: evaluating programs,projects, and products’ (Little Brown & Co., 1982, 3rd edn.)

[94] Fagan, M.: Design and code inspection to reduce errors in program development. IBM System Journal, Vol. 5., no. 3. (1976) 182-211