Software Engineering 2014, 4(1): 19-24
DOI: 10.5923/j.se.20140401.03

Automated Visual Software Inspection System: Re-Making the Fagan Methodology

Oladipo Onaolapo Francisca[*], Ugoh Geraldine Ebere

Computer Science Department, Nnamdi Azikiwe University, Awka, Nigeria

Abstract Software inspection is aimed at detecting error early during the software development process and improving the skills of developers. There are several inspection models for both large and small scale software projects but this paper recognised that they are mostly designed in and for developing countries; in addition it was observed that software inspection in small student groups in Nigeria institutions are based on the traditional, meeting-oriented approach. This therefore necessitated a need to formalize an inspection model suitable for small software projects executed by students in a typical university computer science laboratory. Based on extensive research and analysis, a visual software inspection model is proposed in this paper. This model matured into an inspection tool developed using the techniques of Structured Systems Analysis and Design Methodology (SSADM) and scripting tools. An experimental evaluation of the tool using five study criteria showed that the inspection model was a well-defined disciplined process for the analysis and monitoring of a software development process for a systematic detection of any deviation from the pre-defined specifications of the software system.

Keywords Software inspection, Maintenance, Visual model, Inspection tools

1. Introduction

A software inspection methodology is considered efficient if it meets the primary goal of detecting errors and defects before the beginning of the testing phase in the software life cycle. This way, it contributes in no small measure to improving the overall quality of software corollary budget and time benefits [1]. Reference [2] believed that despite the widespread adoption and success of software inspection, many software products continue to be released with large numbers of flaws. They partly attributed this to the inherent complexity of software systems. The complexity of the software thwarts manual attempts to comprehend it.

It is observed that software design is a complex process due to a number of reasons: the complexity of the problem domain, difficulties in capturing the system's requirements, contradictory and changing requirements, difficulties in managing the development process, difficulties in predicting the final system behaviour or even the behaviour as the system evolves, and so on [3].

As a result of my experiences in teaching software construction and related courses to undergraduate and graduate students in a Nigerian university, and mentoring young software entrepreneurs, I observed that there were basically no formal/automated inspecting tools being adopted even when pair programming approaches are adopted. A great deal of time is spent on correcting errors during the testing phase and most maintenance being done are corrective not adaptive. Something a well structured software inspection tool would have prevented if deployed as errors would have been detected at early phases of development.

The aim of this research is to propose a visual inspection model suitable for small to medium size software projects typically for pupil developers and young software entrepreneurs. This inspection system is built on the traditional Fagan inspection model and it was validated by the development and evaluation of an inspection software tool that will assist inspectors to improve on their skills and productivity; making the inspection interesting and technical. This research is significant because, software inspection and its technical review detects error early in software development cycle when the error is not much and developers skills are improved as a result of the technical review and software inspection participation.

2. Review of Related Research

A taxonomy on Life-Cycle Centric Software Inspection models were carried out by [1]. The goal of the research was to portray the status of research and practice as published in available software inspection publications from a lifecycle angle and present the facts as reported in the literature. The authors in the work; performed an extensive literature survey including a wide source of publications which included

* Corresponding author:
of.oladipo@unizik.edu.ng (Oladipo Onaolapo Francisca)
Published online at http://journal.sapub.org/se

existing surveys at that time. Broadly speaking, they summarized existing survey as follows: Survey by [5] which presented a framework for software development; technical reviews including software inspection [6, 7], and Yourdon's structured walkthrough [8]. The work found that the authors segmented the framework according to aims and benefits of reviews, human elements, review process, review outputs, and other matters. The taxonomy by [1] however, was centered on five primary dimensions: technical, managerial, organizational, economics, and tools. These were used to attempt to characterize the nature of software inspection.

Another review effort focusing on the software inspection process in the light of Fagan's inspection was conducted by [9]. The work summarized and reviewed other types of software inspection processes that have emerged in the last 25 years and also addressed important issues related to the inspection process and examined experimental studies and their findings that are of interest with the purpose of identifying future avenues of research in software inspection.

An inspection model that dispenses totally with the need for the inspectors to be in the same place at the same time was presented by [10]. The model was asynchronous as it replaced the meeting with further individual inspections combined with asynchronous communication between inspectors. A prototype tool that used electronic mail for communication was developed to implement this asynchronous model. The use of electronic mail differentiated this framework from a previously developed asynchronous inspection tool. The inspection model was evaluated in comparison with the traditional, meeting-oriented approach on a number of criteria. The authors made an initial attempt to gain quantitative data by carrying out a small-scale experiment, and whilst encouraging results being obtained, they believed that the number of subjects was too low for any significant conclusions to be drawn and planned for a larger scale experiments are planned in the future in order to obtain more data.

A peer review approach to software inspection was presented by [11]. The paper discussed the software inspection process as a particular type of peer review process and elaborated the differences between software inspection, walkthroughs, and other peer review processes. Reference [12] presented a framework for formal technical reviews (FTR) including objective, collaboration, roles, synchronicity, technique, and entry/exit-criteria as dimensions. The framework aimed at determining the similarities and differences between the review process of different FTR methods, as well as to identify potential review success factors.

Reference [13] differentiated between personal reviews, walkthroughs, and software inspections. She proposed the Pair programming approach as an alternative inspection approach to reviews because the work observed in the course of the research that: Developers simply do not believe that the reviews are worth their time because they have deadlines to meet, or as a result of ego problems. Developers might not want their mistakes being viewed by others or others simply find inspection boring.

An evaluation of computer supported software inspection was presented in a state of the art paper by [14]. The work reviewed several models and issues surrounding software inspection and concluded that software inspection is an effective methodology for managing defects in software development, and that the concept consists of a number of basic steps that has been widely practiced and standardized. The author revealed that the process emerged from project management and product quality requirement perspectives and that managing defects applies to not only source code artefacts but also any other materials in the software development life cycle. The paper also pointed out that Inspections are team-effort activities that bring together a variety of participants who engage as special roles.

3. Materials and Methods

The techniques of the Structured System Analysis and Design Method (SSADM) [15] were deployed in this work to analyse a baseline methodology, the Fagan inspection method. The analysis revealed that the Fagan model is focused on finding defects in the documentations of the development process of software thereby necessitating our proposing and developing a high-level model for a visual inspection method that encourages a straight forward way to establish a supporting system for visual meeting together with the inspection tools.

The web pages were scripted using HTML tags and JavaScript, while the server side programming was developed using the PHP. The open source MySQL Server Database was deployed as the DBMS and the specification tables were tables to handle tasks, roles, projects, meeting attendance, messages, settings, and log.

An experimental comparison of the system was carried out using selected undergraduate students of software engineering from the Computer Science Department of a Nigerian university who had been using manual software testing/inspection procedure. Questionnaires were distributed to the participants and they were required to detail their usability experiences with respect to the tool. A total of 20 students who had used the tool and documented their experiences participated in the evaluation which involved comparing the manual procedure and the automated tool.

Each student was given a questionnaire containing a total of ten survey questions and were required to compare their software construction experiences in five study areas of error detection, time management, corrective maintenance, and team work.

4. Results and Discussion

The motivation for this work was the need for a suitable software inspection system for small to medium sized software project, typically for students of software construction and for young entrepreneurs. Studies by [4] showed that Nigeria's software engineering industry is currently experiencing a dearth of inspectors because the process is views as an expensive one and the practitioners are operating on a budget. In addition, due to time constraints, most individual programmers find it difficult to inspect their products.

The system developed in this work is a suite of tools that exchange information with other tools through a number of connection points. The tools enable the efficient running of the software inspection process independent of time and place, online recording of matters and data management achieved through the network tool. The system provides a flexible process that supports tolerable adoption and method acceptance.

Considering traditional inspection; how large and time consuming it is in arranging for team members that are distributed geographically and are involved in other projects at the same time, to overcome the problem of time and place for inspection meeting to take place; a visual inspection tool is required, where face to face meeting of inspectors are less or not effective. This provides a justification for the model.

Below is the High Level Model (HLM) of the proposed system (Fig 1).

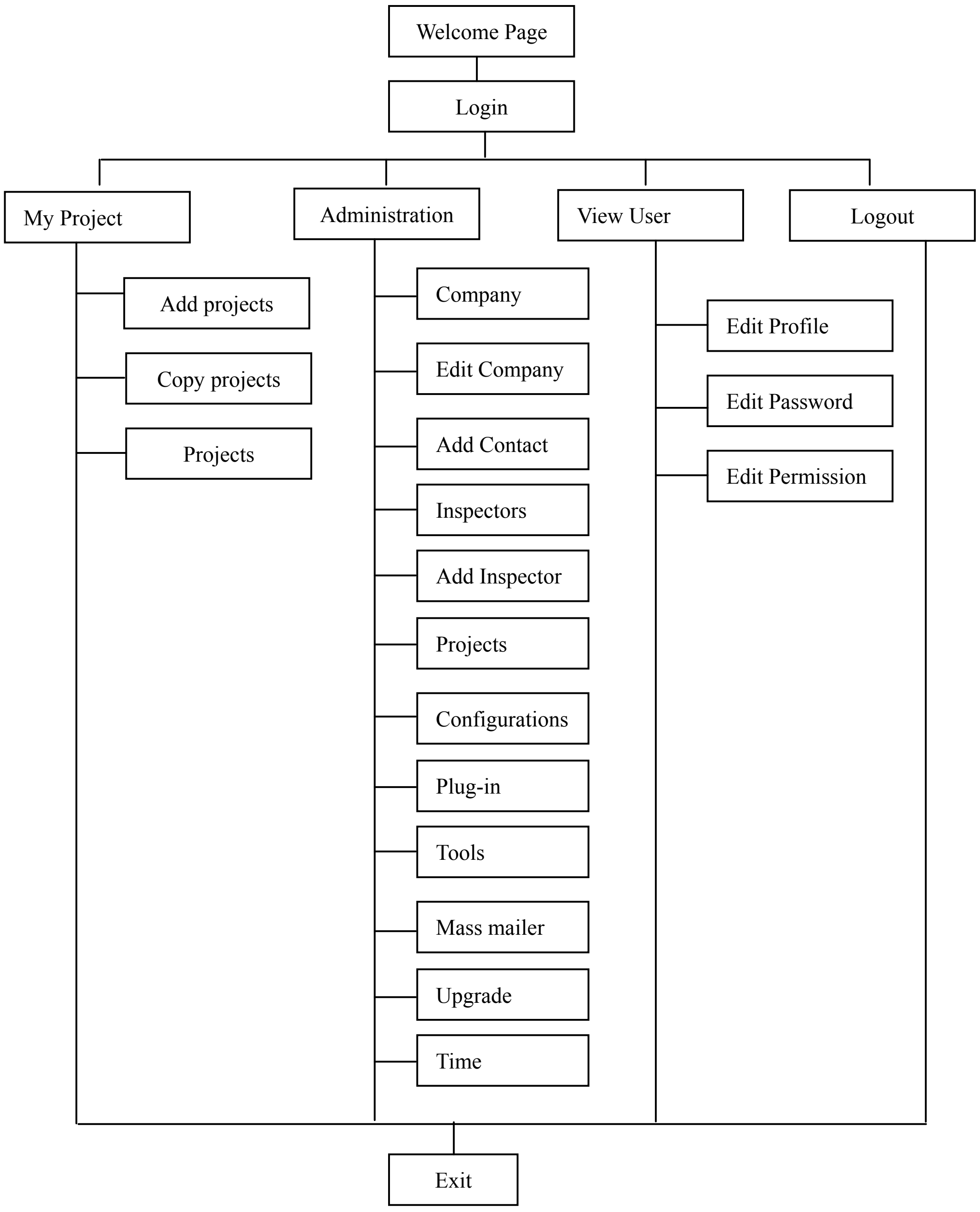


Figure 1. The HLM of the proposed system

System design is an important stage that requires considerable creativity to make the necessary changes in the existing system with view of coming up with a new system that is acceptable to user and easy to manipulate. The HLM is decomposed into four modules, each of which is subdivided into activities to be performed during the inspection activity.

The design was implemented in an inspection software that served the following purposes.

i. Assisting developers in improving their skills and productivity thereby reducing the time spent on testing and debugging.
ii. The system enabled developers exploit all the emerging inspection of technical review opportunities around. This in turn will enhance the creation of standard to control high quality of software.
iii. Making inspection attractive to young developers thereby promoting entrepreneurship.

The implementation of the HLM in a GUI-based application is discussed below.

The users can access the system by logging in through the login page (Fig. 2). This prevents unauthorised users from gaining access to the system.

In addition, the user administration form enables the system administrator to monitor the activities of users of the system. The administrator manages the schema and sub-schema through this page (Fig 3).

The system also involves a software project management toolset (Fig. 4). The page contains all the tools and commands required to manage all the available human and material resources available to properly complete the software project on time and within budget.

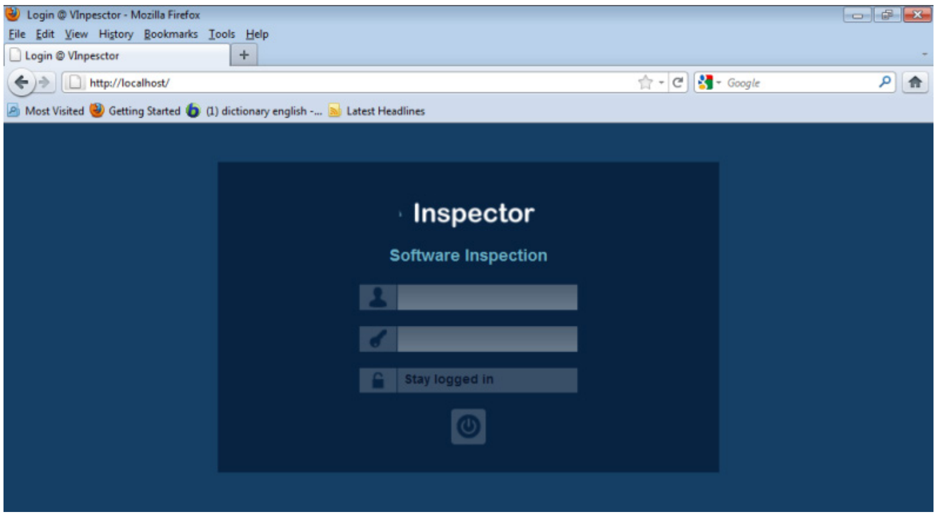
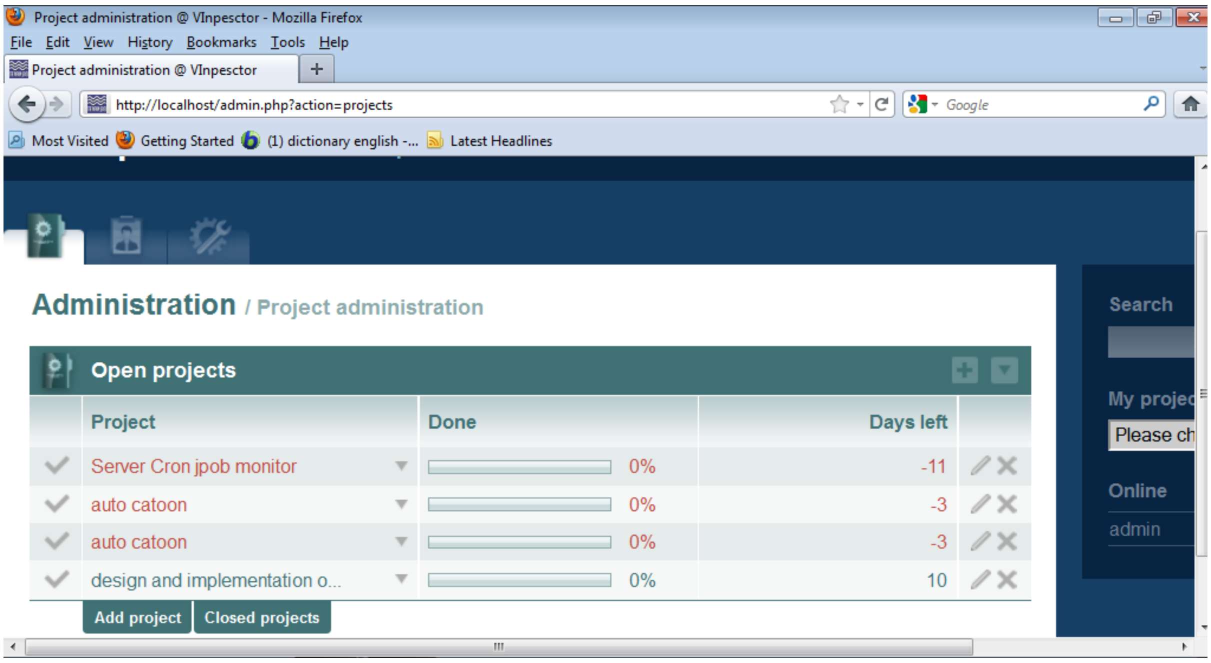

Login @ VInpescteor - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Login @ VInpescteor
http://localhost/
Most Visited Getting Started (1) dictionary english -... Latest Headlines
Inspector
Software Inspection
Stay logged in

Figure 2. Login page

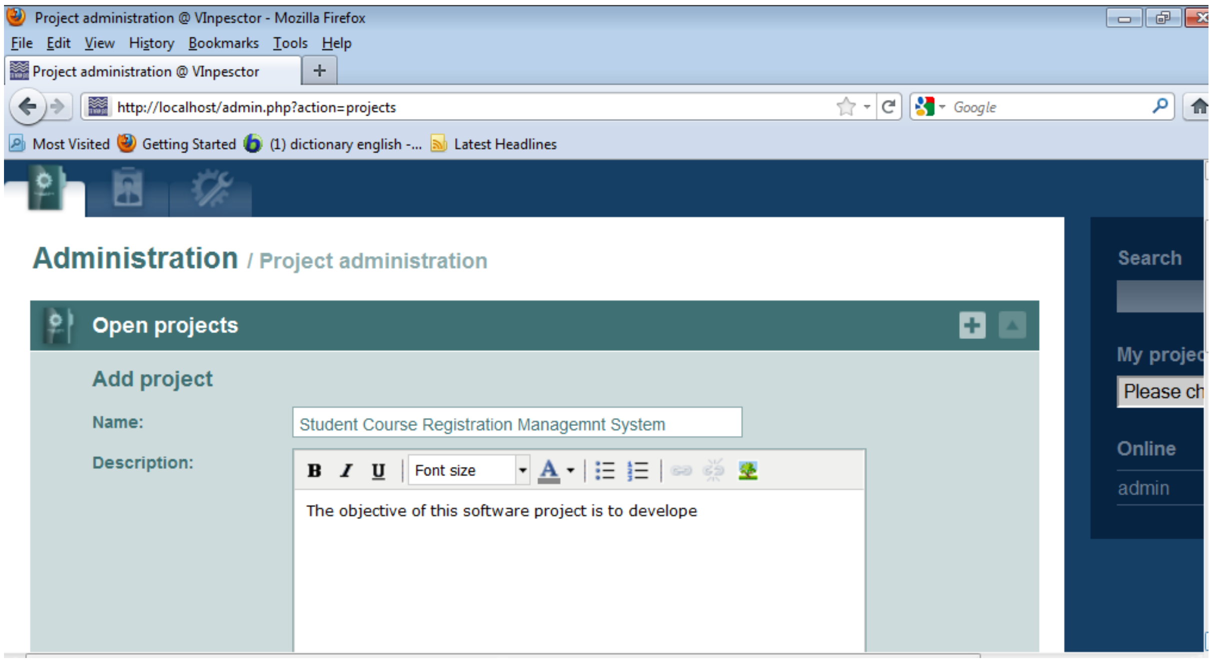

Project administration @ VInpescteor - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/admin.php?action=projects
Most Visited Getting Started (1) dictionary english -... Latest Headlines
Search
My projec
Please ch
Online
admin
Add project
Closed projects

Administration / Project administration

Open projects

Project	Done		Days left
Server Cron jpob monitor		0%	-11
auto catoon		0%	-3
auto catoon		0%	-3
design and implementation o...		0%	10

Figure 3. Administration page for monitoring system progress


Project administration @ VInpesctor - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Project administration @ VInpesctor
http://localhost/admin.php?action=projects
Most Visited
Getting Started
(1) dictionary english -...
Latest Headlines
Search
My projec
Please ch
Online
admin
Font size

Administration / Project administration

Open projects

Add project

Name: Student Course Registration Managemnt System

Description: The objective of this software project is to develope

Figure 4. Project Management page –Add Project

The user interface consists of controls, forms, sub-menus and menus that make navigation through the program easy. Each operation on the menu is activated by the click events of the main option that bears it.

The parallel changeover procedure is advocated in the adoption of the new system such that the whole new system would be run simultaneously with the old system over a period of time. This is suggested in order to compare the output performances and effect manual adjustment where necessary. The advantage is that, if the real system fails, less harm are done to the organization, its disadvantages are the high cost of running both systems and lack of hardware experiences.

4.1. System Evaluation

An analysis of the responses of the 20 users based on the five study criteria are presented below:

a. Early detection of errors

One question was asked on error detection and this was designed to test the ability of tool support to aid faster error detection than manual inspection. All the respondents answered this question and Table 1 showed that 70% of the respondents agreed that tool support enabled them detects errors faster than the manual inspection.

Table 1. Error Detection

Response Options	Statistical analysis	% analysis
Strongly Disagree	0	0%
Disagree	0	00%
Neutral	6	30%
Agree	0	0%
Strongly Agree	14	70%
Total	20	100%

Question: Tool support assisted me in detecting errors faster than manually searching for errors

b. Time management

Because the research was aimed at promoting faster software construction, there were a total of three questions on time management.

Table 2 showed than 60% of the students strongly agreed that they were able to produce faster applications using the system.

Table 2. Time management

Response Options	Statistical analysis	% analysis
Strongly Disagree	0	0%
Disagree	2	10%
Neutral	6	20%
Agree	0	10%
Strongly Agree	12	60%
Total	20	100%

c. Corrective Maintenance

Two questions on corrective maintenance were asked in the questionnaire to test the ability of automated removal of residual errors.72% of the students agreed that corrective maintenance is greatly enhanced by automated software construction tools. Further 88% indicated that they would consider using tool support in their future software construction projects.

d. Collaboration and Team Work

Since the application was designed to help young programmers and entrepreneurs in their software construction projects and our students carry out their projects in pairs, a great deal of attention was paid to the system's ability to provide team support. A total of four questions were posed to users on to test the system's ability to foster team spirits in the students. Specifically, the questions tested collaborative thinking support, overall time taken to complete the project, group support and progress monitoring. 90% of the student admitted using the Project Administration tool of the system to monitor the progress of their software projects; the remaining 10% did not use the tool. 60% of those who used the tool agreed that it actually shortened the overall project time but were unable to say by how much while 70% felt that tool support assisted them in building collaborative spirit. The progress monitoring validation was tied to the time questions and 10% of the respondents were neutral as they did not use the administrative tool of the system.

e. Remote Inspection

This section tested the ability of the students to remotely inspect their partners' contribution to the project and monitor the progress of the implementation remotely. As the system had not been hosted on the net as at the time of conducting the survey, 100% of the respondents were neutral on the ability of the tool to provide support for remote inspection.

5. Conclusions

Software inspection is an essential means of software quality assurance. This paper briefly presented the results of building a software inspection system from a re-make of the Fagan inspection process suitable for small to medium-sized software projects. The GUI-based visual inspection tool assisted in monitoring the development process through a systematic detection of any deviation from its pre-defined specifications when used for projects of its intended size. Overall responses from the evaluation showed that tool support significantly shortened the testing time, promoted team spirits in students and young entrepreneurs and completely eliminated corrective maintenance in addition to improving the developers' skills as a result of the technical review and software inspection participation.

REFERENCES

[1] Laitenberger O, DeBaud J. An Encompassing Life-Cycle Centric Survey of Software Inspection (ISERN-98-32). J. Systems and Software Archive. 2000; 50(1): 5-31. Elsevier Science Inc. New York, NY, USA.

[2] Anderson P, Teitelbaum T. Software Inspection Using CodeSurfer. In: Inspection in Software Engineering, 2001. Proceedings First Workshop on, Paris: 1-9.

[3] Zhiming L. Object-Oriented Software Development with UML. UNU/IIST Report No. 259, 2002. Accessed January 2013. Available: http://www.scribd.com/doc/51693789/UML.

[4] Ugoh G. Design and Implementation of a Visual Software Inspection Model for Distributed Software Engineering Projects. Unpublished Thesis in PGD Computer Science. Nnamdi Azikiwe University, Nigeria. Thesis defended, November 2012.

[5] Kim LPW, Sauer C, Jeffery R. A framework for software development technical reviews. Software Quality and Productivity: Theory, Practice, Education and Training. In: Lee M, Barta B, Juliff P, editors. Chapman and Hall, 294-299, IFIP, 1995.

[6] Fagan ME. Design and Code Inspections to Reduce Errors in Program Development. IBM Systems Journal, 1976; 15(3):182–211.

[7] Weinberg GM, Freedman DP. Reviews, Walkthroughs, and Inspections. IEEE Transactions on Software Engineering, 1984; 12(1):68–72.

[8] Yourdon, E. Structured Walkthroughs. 4th ed. N.Y.: Prentice Hal; 1989.

[9] Aurum A, Petersson, H and Wohlin C. State-of-the-Art: Software Inspections after 25 Years. Software Testing Verification and Reliability, 2002;12(3):133-154.

[10] Murphy P, Mille J. A Process for asynchronous software inspection. In: Software Technology and Engineering Practice, 1997. Proceedings, Eighth IEEE International Workshop on [incorporating Computer Aided Software Engineering]. IEEE Computer Society. Pages: 96 – 104.

[11] Wheeler DA, Brykczynski B, Jr RNM. Software Peer Reviews. In: Thayer RH, editor. Software Engineering Project Management. 2nd ed. Los Alamitos: IEEE Computer Society Press; 1997, 454-469.

[12] Tjahjono D. Exploring the effectiveness of formal technical review factor with CSRS, a collaborative software review system. PhD thesis, Department of Information and Computer Science, University of Hawaii; 1996.

[13] Williams L. A (Partial) Introduction to Software Engineering Practices and Methods. NCSU CSC326 Course Pack, 5th ed. 2008-2009s.

[14] Bordin S. Software Inspection and Computer Support. State of the Art Paper from the PhD Thesis, Department of Information Science and Telecommunications, School of Information Sciences, University of Pittsburgh, 1999.

[15] PAȘCU, P. (2010). The Stages of Implementation of the SSADM System in the Government Institutions. Journal of Applied Computer Science & Mathematics, no. 8 (4), Suceava.