A New Model for Software Inspection at the Requirements Analysis and Design Phases of Software Development

Navid Taba and Siew Ow
Department of Software Engineering, University of Malaya, Malaysia

Abstract: *Software inspection models have been remarkable development in over the past four decades, particularly in the field of automatic inspection of software codes and electronic sessions. A small number of improvements have been made in the field of system analysis and design. The amount of using formal inspection model which is based on single check lists and physical or electronic sessions shows the decrease in interest in it. As inspection, in system analysis phase, is a man-cantered issue, inspectors support using electronic tools will lead to higher efficiency of the inspection process. This paper proposes a comprehensive web-based tool aimed to accelerating the inspection process in the early phases of software development. In order to evaluate the efficiency of the proposed tool, two case studies were conducted to inspect the artifacts from six software projects of two software companies. Comparing the statistics related to the defects detected using this tool with those detected using the formal method shows the efficiency of the used tool.*

Keywords: *Software inspection, software test, software engineering improvement, web-based solution, software inspection tool, inspection metrics.*

Received September 2, 2013; accepted September 29, 2014; Published online December 23, 2015

1. Introduction

The software testing methods which are being used in software development life cycle cannot detect all defects [16]. Researchers have shown that when software inspection is conducted in at all phases of software development, many artifacts are uncovered [1]. Therefore, using appropriate and professional methods for inspecting software, can lead to improvement in software.

In the past four decades, several methods have been proposed for inspecting software. Each method has its own domain, advantages and disadvantages. Most of the proposed inspection methods are based on the formal method proposed by Fagan [6]. In this method, inspectors personally inspect the software and the related documents. Participants in each inspection session include the product producer, an announcer, document reviewers, and the chairman of the session.

Improvements made in software inspection process in recent years have made it possible to carry out electronic inspection after than doing it manually. Several software programs have been proposed for inspecting the program codes automatically. Few researches, however, have been conducted to improve the inspection method, especially in the early phases of software development.

Researches show that defects which are not corrected in the early phases of software development are transferred to the later phases and could result in a 100% increase in expenses. In the early phases, most tasks are developmental in nature, and mainly involve human input (human ware). Therefore, software inspectors have crucial roles in every software inspection process [8].

2. Problem Background

According to Suma *et al.* [17], one of the key challenges to the IT industry is in engineering a software product that will have minimum post-deployment defects. Defect detection, especially in requirements analysis and design phases of software development life cycle dramatically reduces the quality costs [13]. Armour [2] stated that inspection is a way to produce high quality software. Tyran [18] stated that a software inspection is one of the most effective ways to promote quality and productivity in software development. He emphasized that the correction of a defect found early in development can result in the reduction of between 10 and 100 times of the cost to remove defects at the later stages [12].

3. Research Methodology

This section presents the methods used to carry out research. It also explains the research activities; the criteria used to evaluate the proposed software inspection model and discuss the internal and external validity to be satisfied. The research in this article aims to propose a new software inspection model, and its efficiency will be compared to the conventional model. The evaluation will be made based on the data

collected through case studies. Figure 1 show the activities involved in this research.

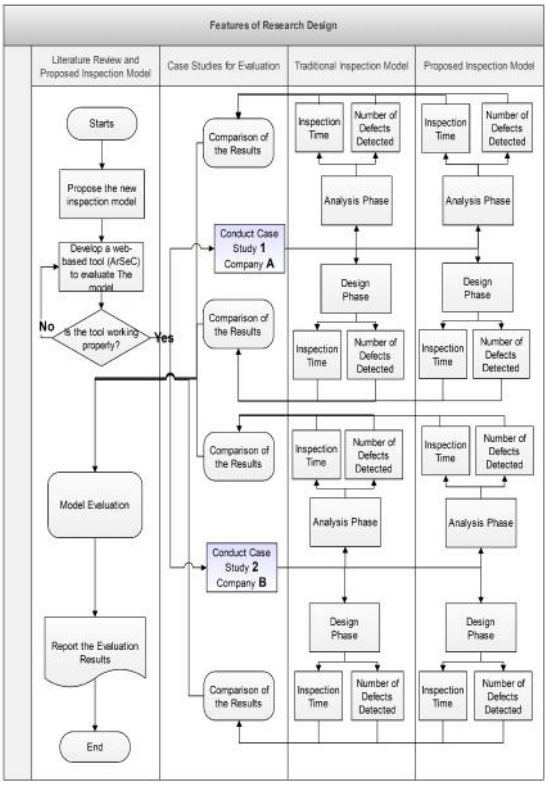

Figure 1. Research design.

A new software inspection model was proposed after conducting a thorough review of the literature pertaining to software inspection. The proposed model was implemented using agile technique in the development of a tool Artifacts and Sessions Control System (ArSeC). ArseC was used to collect data in two case studies. The criteria and metrics used to evaluate the proposed model were also defined for the case studies. The following sections explain the research activities in detail.

3.1. Activity 1: Proposing A New Software Inspection Model

While In the formal software inspection method, the defects detected are not classified, and the causes and effects of recognized defects are not documented in a structured manner [9]. A new and more efficient software inspection model is proposed to overcome the shortcomings of existing formal software inspection methods.

The design of the propose model is based on the recommendations made by prominent researchers in the area of software inspection. The proposed model is designed to detect and remove the defects in the first two phases of software development the requirements analysis phase and the design phase. The main reason for doing this is that defects in these two phases, if not detected and removed, could adversely affect the later phases of software development, especially in the coding phase. Another reason is that existing software inspection models cannot detect defects from the documents of these two phases [4].

The proposed model provides a database to store and keep track of defects found in the two phases. Thus, more defects can be detected as compared to the formal software inspection model. It can detect defects in the atrifacts in the requirements analysis phase and the design phase regardless of the type of development method, tools, and technologies used.

The artifacts in the requirements analysis phase are usually the specification documents of the system [15]. In the design phase, the artifacts include the design specifications together with various design graphs and diagrams. The most significant feature of the proposed inspection model is that defects detected can be classified and stored in a database, and the information can be accessed using ArSeC. Inspectors can inspect the artifacts by focusing on the list of classified defects using checklists generated by ArSec. The database also includes the causes and effects of a defect, and this facilitates the software inspection process as well as the Defect Removal (DR) process. The ability to add new defects into the classified defects database makes the database dynamic and up-to-date.

The database, which also stores frequently encountered defects, can be very useful to software inspectors to help them to detect defects easily and quickly. As ArSeC generates the inspection checklists and the defects found in the inspected software are also recorded using the tool, this reduces the inspection time and at the same time increases the productivity of competency of the inspectors.

3.2. Activity 2: Developing A Tool for Implementing the Proposed Model

The proposed software inspection tool, ArSeC can facilitate the software inspection process and improve the productivity of the inspectors. ArSec was developed using agile development technique under the .NET environment.

3.3. Activity 3: Evaluating the Proposed Software Inspection Model

To evaluate the proposed model, two case studies were conducted in two companies. The first case study conducted in a software company and the second one in an industrial company. Six software projects were inspected in each of the two companies. In each project, 48 artifacts in the requirements analysis phase and the design phase were inspected. The formal inspection process was first used to find the defects in the artifacts. The defects detected and the inspection times taken were recorded using ArSeC. The inspection process was then repeated using the proposed inspection model. Similarly, the inspection

data were collected and recorded using ArSeC. These two sets of data were compared to determine which of the two models is better at finding defects, reducing the inspection time and increasing the productivity of the inspectors.

To avoid bias and to ensure validity and reliability of the research, both inspections were done by the same group of three inspectors under the same environment and on the same artifacts. These inspection processes were repeated by another group of inspectors. This approach is used to prevent any bias that could result because of knowledge and experiences in software inspection among the software inspectors. Figure 2 shows the evaluation process of the proposed software inspection model.

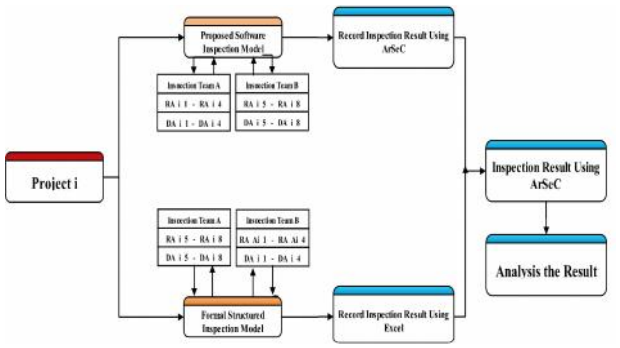

Figure 2. Research roadmap.

4. Internal and External Validity

In any academic research, it is crucial to control the intervening variables in the results to make better interpretations [3]. Thus, the conditions must be prepared to ensure that different factors cannot threaten the internal and external validity. In this way, the researcher will be able to prevent or minimize adverse impact caused by any unexpected factors. The most significant factor in internal validity is compatibility and sustainability tools to measure the metrics. Using ArSeC, a common database is provided and this eliminates the need for this intervening variable. Also, having trained inspectors to conduct the case studies will ensure the internal validity of the study. Common artifacts whish are inspected by formal and proposed model support internal validity from another aspect. Finally, the case studies conducted the a same period time and the same conditions, so the time factor which might impact on the results.

The findings of the research can be generalized, if more case studies and more software projects are considered. This invariably involves more different more settings and to a larger number of variables, thereby, the research will have higher external validity. The Implementation of the proposed model in the different projects satisfies the external validity of the research. The external validity of the study can be further enhanced by choosing different artifacts in each project.

5. Pioneer Inspection Model

The proposed software inspection model performs defects removal as an important task of inspection. It also exploits the capabilities of a collaborative system ArSeC. There is continuous process improvement due to the creation of swap iteration in the inspection model kernel. Making and modifying some rules related to defects, adds intelligence and learning capabilities to the model. In order to validate the model, it is implemented in a real software inspection project. The model consists of four important phases preparation; defect plan design; generative inspection procedures; and inspection process evaluation as shown in Figure 3.

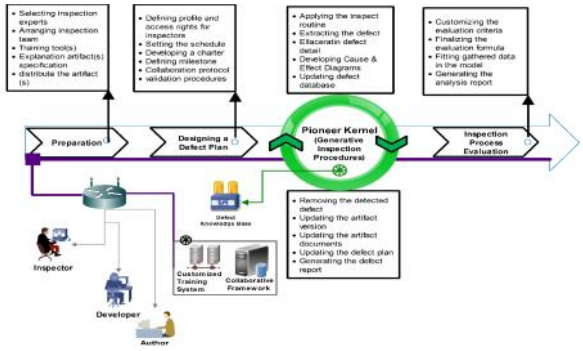

Figure 3. Pioneer inspection model.

To execute this conceptual model, inspectors, developers and users have to communicate during the process using a proper tool. Also, those involved in the inspection process should have good technical knowledge and expertise on tools, methods, and inspection of artifacts. In order to assess the effects of defects property, the model suggests designing and using a comprehensive database. The phases of the aforementioned model are explained as below.

5.1. Preparation

The initial stage of the inspection process involves preparation of the environment, and appointing the inspectors. In this phase, the inspectors are selected based on the required skills, and the specifications of the inspected artifacts. The inspection team and the team organization structure are then arranged in a centralized, decentralized or distributed structure. The responsibilities, roles, and duties of each member of the team are also identified. Following this, the specifications and, in some cases, the artifacts are distributed among team members or to the inspection team. Finally, quick flash test is conducted to ensure that the inspectors are conversant with the situations, and knowledgeable to respond accordingly.

5.2. Designing Defect Plan

Proper resource allocation, scheduling and goals determination are crucial to the success of an

inspection process [7]. In this phase, the first step is to define the profile and access right of the inspectors. The next steps Include defining an appropriate schedule, and a complete charter that is includes provisions for: Collaboration method, resolving possible disagreements, determining milestones and, collaboration protocols.

5.3. Generative Inspection Procedures

This phase involves repeating of the two complementary sets:

5.3.1. Detect Diagnosis

The first set contains the required actions to identify and recognize the defects. The main functions of Detect Diagnosis (DD) are performing inspection procedures; defect detection; explaining the details of each defect; sketching the cause and effect diagrams and updating the defects databases.

5.3.2. Defect Removal

DR is the second routine set, and it is considered the most specific attribute of the proposed model and makes the model intelligent and generative, and it also satisfies the main goal of the inspection process, which is defect removal. The other supplementary functions of this phase are: Removing defects from artifacts and preparing a new version; updating related documents; defect plan; and creating defect report.

5.4. DD and DR Swapping

As mentioned above, the two sets that constitute intelligent inspection must be run iteratively and periodically. The iterative execution feature makes it possible to remove newly-arisen defects, while detecting other defects. The key factor is to recognize when the cycle should be broken, and when to enter into the last phase. However, this should be considered in the defect plan as certificate instruction and termination criteria.

5.5. Defect Database

The actions of the two sets, DD and DR, are carried out by using a database that contains the related rules and facts of defects. The potential defects and their causes are stored in this database, and when a defect is detected, the inspectors establish, reform or modify the rules. The use an inference engine may help the inspectors in their tasks. This inference engine could alert the inspectors about the possible defects and shows the possible causes (if any defect is found).

5.6. Inspection Process Evaluation

As there is a specific plan for each inspection, the evaluation process should be done according to the plan. Thus, the first step of the last phase is to customize the evaluation metrics. The second step is to finalize the evaluation formulas according to pre-defined criteria. The next step is to put data into the related formula and to analyse them. The results of these evaluations can be useful for future inspection plan designing and for improving the methods used in evaluation. It adds the learning feature to the system, which is the special attribute of an intelligent model.

5.7. Involved Professionals in Inspection Model

Users, software developers, independent and internal inspectors are the people involved in the inspection process. The use of web-based distributed tools and collaboration framework not only facilitates the inspection process, but also prevents gaps and overlaps in the task carried out. Another advantage of using this kind of environments is that it involves the inspection process employers from different time zones, different geographic locations. Finally, it can be said that the integrated environments supported by a relational or networking database are more suitable for conducting the experiments of the projects, when compared to some traditional document-based approaches.

6. Comparison the Proposed Model With the Current Similar Models

The features and characteristics of the current software inspection model are compared with the capabilities of the proposed model as shown in Table 1. None of previous model has a systematic policy, mechanism, or procedure to provide learning for the model. Putting the finding the defects with corresponded cause and effect schema, provide learning to the system that after each execution empowers the inspectors through providing more classes, instances and cause-effect relations.

Table 1. Comparison of the proposed model features with the current models.

Inspection Method	Team Size	Multiple Sessions	Meeting	Detection Method	Learning
Proposed Model	**Small**	✓	✓	**Dynamic Checklist**	✓
Two-Person	Small	X	✓	Ad-hoc	X
Ftarm	Large	X	Opt.	Ad-hoc / Checklist	X
Phased	Small	✓	✓	Ad-hoc	X
Gilb	Large	X	✓	Checklist	X
Verification Based Inspection	Small	✓	✓	Reading	X
Structured Walkthroughs	Large	X	✓	Ad-hoc	X
Fagan	Large	X	✓	Ad-hoc	X
N-Fold	Small	✓	✓	Ad-hoc	X
Meeting-Less	Large	X	X	Variety	X
Simplified Software Ins. Process	Small	X	✓	Checklist	X

7. ArSeC an Automated Inspection Tool for DD

ArSec is an automated software inspection system was developed to detect potential defects in the early phases of software development, as well as to facilitate the process of recording the defects. It has a database to store defects frequently detected in the requirements

analysis phase and design phase. The database also, stores information on the potential causes of each defect, thus preventing such defects from occurring. Figure 4 shows a snapshot of the ArSeC tool in accelerating and supporting the inspection process of software. In addition to, defining the phases of each project, this tool provides facility for classifying the defects. The tool can also generate various checklists that are related to inspection of each artifact.

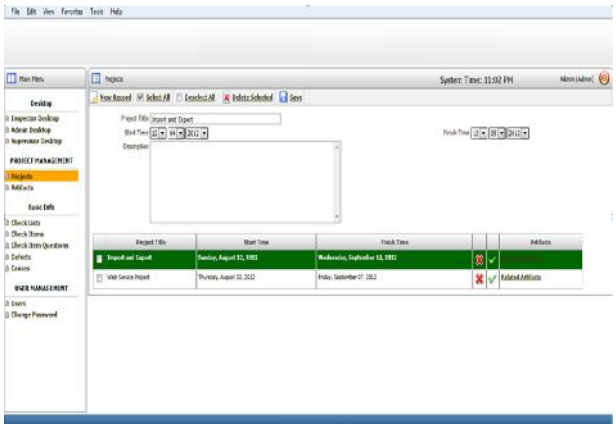

Figure 4. Web-based software inspection tool ArSeC.

Some capabilities of the ArSeC is presented in Table 2. The capabilities and services of the ArSeC compared with the characteristics and capabilities of the other inspection tools [11]. Although, "sending emails" is supported by most of current inspection tools, only compass and EMS, plus ARSeC provides scheduling as an important feature for managing the inspection sessions. The most important point here is that except ArSeC that is designed to work under web, only two inspection tools are web-based and other at last work on a network. Authorization and authentication is another unique feature of ArSeC. The inspection in ArSeC is assumed as a formal process. Therefore, every user despite of his role have to be defined by an authorized person and investigated for validation through an authentication process includes passcodes (changes regularly), security questions and IP checking. Automated analysis is another important service that is available in ArSeC and before is presented by CSI and ICICLE inspection tools before in a limited form. ArSeC analyses the corresponded inspection data and draw a chain of cause and effect relations in form of a graph. Although, voting facilities are provided by most of current inspection tools, for the first time weighted voting is supported by ArSeC. In a fuzzy logic schema, each get a rank based on his experiences s well as skillfulness. So weighted voting could be assumed as a supporting for such a schema. Finally, only EMS inspection supports an inspection process among current inspection tools, while ArSeC, developed to support the Pioneer inspection process and deliver the necessity requirements of that.

Table 2. Comparison of ArSeC capabilities over current inspection tools.

Capabilities	Inspection Tool									
	Arsec	Compass	WIP	EMS	ICICLE	Inspeq	CSI	Scrutiny	Inspecta	Hypercode
Scheduling Support	✓	✓	X	✓	X	X	X	X	X	X
Web-Based	✓	X	✓	X	X	X	X	X	X	✓
Distributed Meeting	✓	X	X	✓	X	X	✓	✓	X	X
Defect Classification	✓	X	✓	✓	✓	X	✓	✓	X	X
Checklists	✓	X	✓	X	X	✓	X	X	✓	X
Data Collection	✓	✓	X	X	✓	X	✓	✓	X	X
Automated Analysis	✓	X	X	X	✓	X	✓	X	X	X
Weighted Voting	✓	X	X	X	X	X	X	X	X	X
Process Support	✓	X	X	✓	X	X	X	X	X	X
Synchronous Facility	✓	X	X	X	✓	X	✓	✓	X	X
Authorization and Authentication	✓	X	X	X	X	X	X	X	X	X

8. Case Studies

Artifacts in the requirements analysis phase and design phase of six software projects of two different companies were inspected using ArSeC tool. Two groups of inspectors with similar expertise and experience conducted the inspections.

Researchers recommended a competence evaluation test to validate the experiments [14]. Therefore, a competence evaluation test was conducted to confirm the sameness of the two groups in terms of their individual capabilities, experiences or artifact type do not affect the findings of the inspection. Following tasks ensures the competence evaluation test:

- *Criteria* 1: Some artifacts of each project were formally inspected by group A and some by group B.
- *Criteria* 2: Some artifacts of each project were inspected, using the ArSeC, by group A and some by group B.
- *Criteria* 3: Chairmen of groups A and B were continually swapped.

8.1. Data Collection

Table 3 shows the data collected from the two conducted case studies. This data was collected from inspecting 84 artifacts in two software projects. The selected artifacts were the requirement forms including information required for system analysis obtained using fact-finding techniques such as interviews, observation, questionnaires and review recordings. The Design phase artifacts were the UML diagrams. The most important defects found by the inspectors in the requirement analysis phase and the design phases relate to the system specifications.

Forty eight artifacts were inspected formally by group A and 48 artifacts were inspected by the same group using ArSeC tool. Group B also inspected 48 artifacts using the formal inspection method, and 48 using the ArSeC tool. Of the 96 artifacts inspected, 16 are related to each project from A to F. Projects A, B, and C are conducted in company A and projects D, E and F are done in company B.

Table 3. Data collected from two conducted case studies.

Inspected Artifact Case Study I	No. of Defects Found by New Tool (Team A)	No. of Defects Found by Formal Method (Team B)	Effective Percentage	Inspected Artifact	No. of Defects Found by New Tool(Team A)	No. of Defects Found by Formal Method (Team B)	Effective Percentage
Artifact No. 1	10	7	42%	Artifact No. 7	7	6	16%
Artifact No. 2	3	2	50%	Artifact No. 8	4	5	-20%
Artifact No. 3	4	4	0%	Artifact No. 9	2	2	0%
Artifact No. 4	9	5	80%	Artifact No. 10	6	3	100%
Artifact No. 5	6	4	50%	Artifact No. 11	5	3	66%
Artifact No. 6	11	7	57%	Artifact No. 12	7	5	40%
Total	43	29	48%	Total	31	24	29%
Inspected Artifact Case Study II	**No. of Defects Found by New Tool (Team A)**	**No. of Defects Found by Formal Method (Team B)**	**Effective Percentage**	**Inspected Artifact**	**No. of Defects Found by New Tool (Team A)**	**No. of Defects Found by Formal Method (Team B)**	**Effective Percentage**
Artifact No. 1	14	11	27%	Artifact No. 19	3	3	0%
Artifact No. 2	5	4	25%	Artifact No. 20	6	5	20%
Artifact No. 3	3	4	=25%	Artifact No. 21	10	7	43%
Artifact No. 4	4	3	25%	Artifact No. 22	2	2	0%
Artifact No. 5	8	6	33%	Artifact No. 23	4	3	33%
Artifact No. 6	13	11	18%	Artifact No. 24	9	7	28%
Total	47	39	20%	Total	34	27	26%

8.2. Discussion and Results

As it shown in Table 4, the number of defects detected by group A using the formal inspection method is less than 60% of the artifacts detected by group B using the proposed tool, ArSeC. The number of defects detected by group B using the formal method is less than 52% of the artifacts detected by group A using the proposed ArSeC tool. Therefore, the first finding can be attributed to the similar level of capabilities of the two groups of inspectors.

Table 4. Comparison of the proposed model efficiency with the formal model.

Case Study	Detected Defects Found by Formal Method	Inspection Efforts*	Efficiency Rate**	Defects Found by New tool	Inspection Efforts*	Efficiency Rate**	Efficiency Improvement
A	53	1855	35	74	1702	23	**35%**
B	66	2046	31	81	1539	19	**47%**
* Person per Minute ** Minutes per Defect							

The results show that by using ArSeC tool, the inspectors were able to detect the defects faster. The defect detection is 35 minutes for each defect for group A using formal inspection, but 23 minutes for each defect for the same group using the ArSeC tool. These Figures show a 37% improvement in the defect detection rate when ArSeC is used for inspection.

Defect detection rate is 31 minutes for each defect for group B using formal inspection. This shows 19 minutes saving for finding each defect, which means a 40% improvement. It is clear that group B is less efficient in detecting defects. However, considering the overall performance of both groups and using both methods, this difference in the detection rate is not significant and thus, has no impact on the research findings as shown in Figure 5.

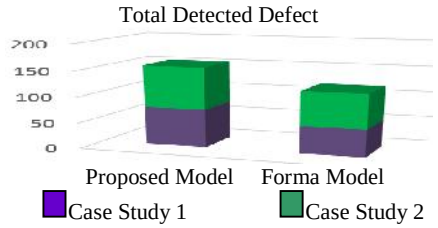

Figure 5. Efficiency comparison of formal inspection and the proposed inspection model.

9. Conclusions

Defects those are not detected in the requirements analysis phase and the design phase of software development, will impact on the quality of the software system and even lead to software failure [5]. This research proposed a software inspection model with the aim of improving the inspection process. This inspection model was established based on the good features found in other software inspection models. The key focus of this proposed model is on inspecting artifacts in the requirements analysis phase and the design phase of software development.

The proposed model was evaluated by conducting case studies conducted in a software company and an industrial company. The data to be used in the evaluation process was collected from six projects [10]. The evaluation was based on number of defects detected in each artifact; defect density; inspection time; defect finding efficiency; and inspection rate.

One of the objectives of this research is to give technical and professional support to software inspectors by providing a comprehensive web-based tool. Potential DD in previous projects can assist inspectors in inspecting artifacts. Recording the information from any inspection minimizes human errors and avoids repetition. A tool, ArSeC, was

developed to support the model and to facilitate the inspection process. The findings of the case studies show that ArSeC helps to make the software inspection process more efficient. The defects database is a very useful feature of ArSeC. As more defects detected in every project, is added to the database, it becomes more comprehensive, and the efficiency of software inspection process will improve from one project to the next.

Acknowledgment

This research was funded by the University of Malaya under the Postgraduate Research Grant (PPP), Account Number: PS027-2012A.

References

[1] Agrawal M. and Chari K., "Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects," *IEEE Transactions on Software Engineering*, vol. 33, no. 3, pp. 145-156, 2007.

[2] Armour P., "The Unconscious Art of Software Testing," *Communications of the ACM*, vol. 48, no. 1, pp. 15-18, 2005.

[3] Creswell J., *Educational Research Planning, Conducting, and Evaluating Quantitative and Qualitative Research*,. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2008.

[4] Dumsmore M., Roper M., and Wood M., "The Development and Evaluation of Three Diverse Techniques for Object-Oriented Code Inspection," *IEEE Transaction on Software Engineering*, vol. 29, no. 8, pp. 677-686 , 2003.

[5] Elgammal A., Turetken O., van den Heuvel W. -J., and Papazoglou M., "Formalizing and Appling Compliance Patterns for Business Process Compliance," *Software and Systems Modelling*, vol 15, no. 1, pp. 119-146, 2014.

[6] Fagan M., "Design and Code Inspections to Reduce Errors in Program Development," *IBM System Journal*, vol. 15, no. 3, pp. 182-211, 1976.

[7] Nagpal G., Uddin M., and Kaur A., "Grey Relational Effort Analysis Technique using Regression Methods for Software Estimation," *The International Arab Journal of Information Technology*, vol 11. no. 5, pp. 437-445, 2014.

[8] Taba N. and Ow S., "Improving Software Quality Using a Defect Management-Oriented (DEMAO) Software Inspection Model," *in proceedings of the 6th Asia Modelling Symposium*, Bali, pp. 46-49, 2012.

[9] Houdek F. Schwinn T., Ernst D., "Defect Detection for Executable Specifications-An Experiment," *International Journal of Software Engineering and Knowledge Engineering*, vol. 12 no. 6, pp. 637, 2002.

[10] Hwang S., "Essential Contents for Software Development Process and Software Quality Education," *International Journal of Engineering Systems Modelling and Simulation*, vol. 6, no. 1-2, pp. 44-53, 2014.

[11] Johnson C., "Forensic Software Engineering: Are Software Failures Symptomatic of Software Problems?," *Safety Science*, vol. 40, no. 9, pp. 835-847, 2002.

[12] Jorgensen M. and Shepperd M., "A Systematic Review of Software Development Cost Estimation Studies," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 33-53, 2007.

[13] Leite J., Doorn J., Hadad G., and Kaplan G., "Scenario Inspections," *Requirements Engineering*, vol. 10, no.1, pp.1-21, 2005.

[14] Mishra D. and Mishra A., "Simplified Software Inspection Process in Compliance with International Standards," *Computer Standards and Interfaces*, vol. 31, no. 4, pp. 763-771, 2009.

[15] Pressman R. and Maxim B *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 2014.

[16] Ramler R., Wolfmaier K., Stauder E., Kossak F., and Natschläger, T.," *Product-Focused Software Process Improvement*, pp. 14-27, 2009.

[17] Suma V., Nair T., and Gopalakrishnan R., "Effective Defect Prevention Approach in Software Process for Achieving Better Quality Levels," *in Proceedings of World Academy of Science: Engineering and Technology*, pp. 288-292, 2008.

[18] Tyran K., "A Software Inspection Exercise for the Systems Analysis and Design Course," *Journal of Information Systems Education*, vol. 17, no. 3, pp. 341-351, 2006.

Navid Taba is PhD candidate at University of Malaya in Software Engineering field, since 2009. He earned his first Doctorate from Phonies University in field of Leadership specialization in Information System and Technology. His areas of interest are software inspection, system analysis, project management, expert system and fuzzy logic.

Siew Ow obtained her PhD degree from the University of Malaya in 2000. She joined the university in 1992. Currently, she is an Associate Professor of the Department of Software Engineering, Faculty of Computer Science and Information Technology. Her research interests include software testing, software metrics, project management, E-learning, computer game development, and health informatics. She has published more than 80 papers in scholarly journals and conference proceedings, locally and at international level.