Machine Learning Based Inspection Technique in Software Inspection Lifecycle

[1]Muaz Gultekin [*2]Ozge Ozturk and [*3]Selin Bircan Tutgun
[1]Yalova University
[* 2]Yalova University
[*3]Yalova University

Abstract

One of the most important elements in software engineering is quality of software. Many standards and tools have developed to improve software quality. Despite this fact, the software still has many quality issues. Until recently used methods to improve the quality of software were test and review techniques. In some cases test and review techniques are not overcome the problems that's why a new solution which's called as inspection is suggested. Inspection is a more effective method than review in many aspects and improve quality of software substantially. In this study we give a brief description of inspection in software engineering and suggested a novel approach in software inspection lifecycle.

Key words: Software lifecycle, quality, inspection, review

1. Introduction

Researchers have responded to problems, that is the complexity of the code, by studying methods of formal correctness verification for programs. There are many approaches to solve this issues, in this study a novel approach developed based on mathematical theorems which is very practical. But we come across many problem while implementing our approach. This method is not acceptable to software because software programs involve long complex expressions and require patience, time, and effort developers do not think that they have [1]. Inspection methods have more advantageous than informal reviews. Many companies support inspection to improve projects they developed. The term "inspection" can be assigned different meanings through software process. According to us, software engineering process is partially ordered activities, with associated roles, input artifacts, output artifacts, and supporting process assets, intended to produce or enhance software products within a specified context.

*Corresponding author: Address: Sogani Mah. Gokce Sok. No : 24 Daire : 5 Bahcelievler / Istannbul Istanbul 09100 Turkey. E-mail address: muaz.gultekin@gmail.com

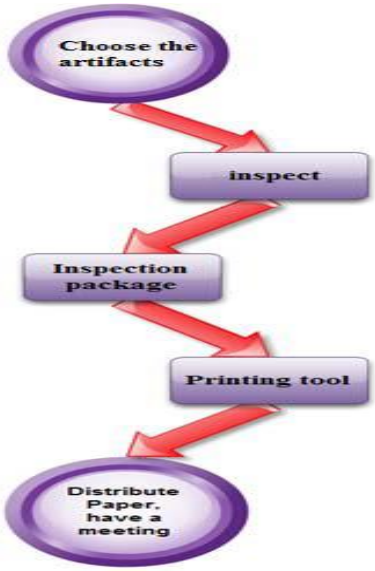

Figure 1. Traditional Inspection Flow

Inspection is the most effective method to locate errors in software to minimize nowadays. [4] Generally, a process that is defined by the document as written form and checkpoint. Formal inspection plays an important role in software quality. Inspection consist of planning, overview meeting, preparation, inspection meeting, causal analysis, rework and follow-up phased .The benefits of inspection process can be summarized as follows: to minimize error rate in the project, time and budget saving [1]. As well as, according to our research showed two common problem in analysis and design phases of the formal inspection. These can be listed as follows: Absence of inspection in a process, especially Inspection meeting, decreases its reliability. Personal who works in office is the important part in inspection process and their not being capable causes inspection without quality.

2. The Proposed Model for Software Inspection

There are two important phases in an inspection - analysis and design - and make transition from one phase to another. We know that SDLC (Software Development Life Cycle). [1] In the traditional inspection has been detected errors and removed but they would not be recorded in any way. We can use defect databases and knowledge databases in proposed model to achieve an intelligent process. We can refer to scenario based model. This model have some abilities like document based, organized, milestones and deliverables are defined in each phase, although has some disadvantages. One of the most important disadvantages of the model is it requires more budget and there is no way to return to the previous step in some cases. According to some researchers, some defects can raise from one phase to the next. Researcher proposes that improving final tests in each phase - especially validation and verification - that are method of avoiding defects to raised process.

We can detect and remove a portion of defects and reduce defects increasing, by inserting inspection to the SDLC model. The scenario based inspection model not only improve software quality also detecting some possible defects. This is a great benefit for software project in risk management.

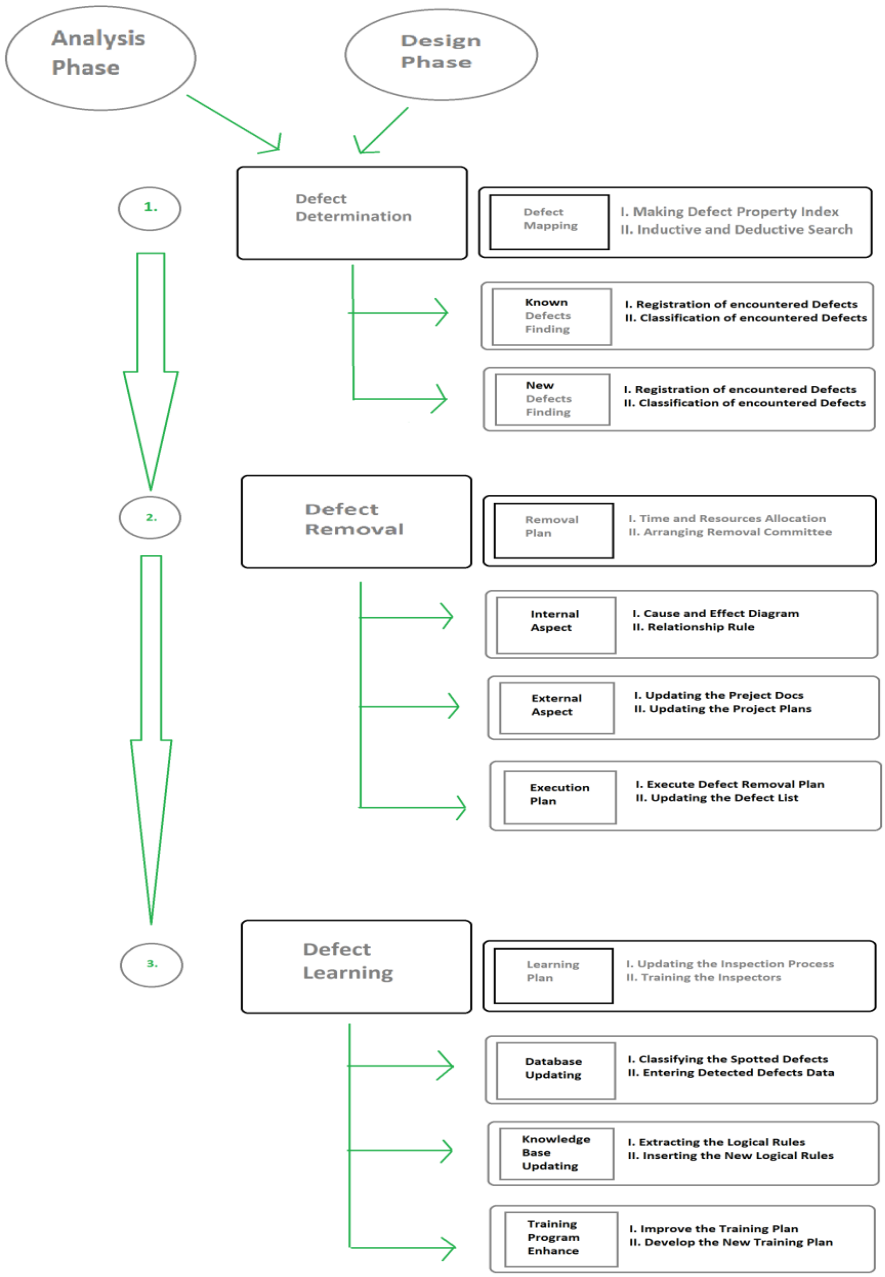

Figure 2. Scenario Based Model in Inspection

2.1. Unique Characteristic of the Proposed Model

The proposed method adapted to SDLC methodology is based on the software development phase. At the end of each of analysis and design phases, the proposed actions are not only documents, deliverable and non-deliverable products but also inspect the software gradually and indirectly. The most important characteristics of this method is that making inspection gradually and step by step of each phases of software development. This inspection approach removes some possible defects in each phase and avoids amplifying these defects in the next phases. This model improves the quality of software during development process. Advantages of gradually inspection are;

- Transition facilities
- Scheduling improvement
- Cause and effect paradigm
- Facilitate risk management

2.2. Proposed Model Steps

There are different activities and some working products in each of analysis and design phases of SDLC besides that there are three steps in the proposed model.

Defect determination: The inspectors learning process starts with preparing the information to possible defects in this step. The substantial action in this step is defect mapping that identifies which defects with which properties are related to what section, location, component, document or artifact. We use inductive and deductive methods to detect the defects. The prevalent methods of gradually inspection artifacts but innovative methods to detect the defects use automated tools, simulation etc.

Defect removal: The proposed model separates from other inspection methods and aim to detect report and remove defects. Removal in this step by taking into account the previous step stored information - which actions are done by whom with which skills and using which instruments and resources in how many times and in which order to clear the defect.

Defect learning: The most important and interesting point of this model is creating an intelligent inspection model. Learning is fundamental factor for an intelligent model. A learning plan is created and executed through which, according to founded results, inspection process is modified and updated in this step. Three following actions of this step are done.

- Database updating: The relations and fields of defects databases are updated by finding some defects.
- Knowledgebase updating: Defects logical rules are extracted and then inserted into the knowledgebase and database updating.
- Training program enhancement: Inspectors training programs, that are preparation programs or defects discovering and registering methods, will be improved.

2.2.1 Defect learning

2.2.1.1. K-Means Algorithm

K-means is one of the simplest unsupervised learning algorithms to solve the clustering problems. The algorithm follows a simple way to classify a given data set through a certain number of clusters. The main objective of this algorithm is to define k centroids for each cluster. These centroids should be located carefully because of different location entails different result. Therefore, the better choice is to place centroids far away from each other possibly. The next step is taking each point belonging to a given data set and associating it to the nearest centroid. When there is no point that waiting to be placed, the first step is completed. At this juncture, we need to re-calculate k new centroids as balance of the clusters resulting from the previous step. Then we have these k new centroids, a new linking has to be done between the same data set points and the nearest new centroids. A loop has been formed in this way. As a result of this loop, the k centroids change their location step by step until no more changes to be done. In other words centroids don't move no longer. The aim of this algorithm to minimize an *objective* function.
The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

Where $\left\| x_i^{(j)} - c_j \right\|^2$ is a selected distance measure between a data point $x_i^{(j)}$ and the cluster center c_j, is a pointer of the distance of the *n* data points from their self-cluster centers. The algorithm is formed of the following steps:

- Place K points in the space by the objects that are currently displayed cluster. These points symbolize first group centroids.
- Assign each object in the group that has the closest focus.
- When all objects have been assigned, recomputed the positions of the focus point of K.
- Repeat Steps 2 and 3 until the focus no longer move. This outputs a separation of the objects into groups from which the metric can be minimized, can be calculated.

Although it can be shown that the method always terminate, K-means algorithm does not necessarily find the optimal configuration corresponding to the global minimum of the objective function. The algorithm is also sensitive to the initial cluster centers randomly. K-means algorithm may be executed multiple times in order to reduce this effect.
K-means has been adapted to many problem domains. It is a good candidate for expansion to work with fuzzy feature vectors. We researched about k-means and these are what I got k-means is one of the simplest algorithm which uses unsupervised learning method to solve known clustering issues. It works really well with large datasets. However, there are also drawbacks of K-Means which are:
Strong sensitivity to outliers and noise
Doesn't work well with non-circular cluster shape -- number of cluster and initial seed value need to be specified beforehand
Low capability to pass the local optimum.

Is there anything great about k-means, because it seems that the drawbacks are beyond the good things about k-Means

3. Results

In this study we analyze 50 project life cycle inspection phase. In table 4.1 a part of database listed.

Table 1 Database Sample

	Error in Design Phase (#man/day)	Error in Analysis Phase (#man/day)	Error in Deploy phase (#man/day)	Excellent	Good	Normal	Bad	Poor
1	51	71	119	×				
2	46	64	122		×			
3	65	77	101	×				
4	93	89	98	×				
5	166	101	143	×				
6	45	67	88				×	
7	98	68	80			×		
8	66	60	84				×	
9	47	89	65		×			
10	92	56	130			×		

4. Conclusion

The developed model give opinion about applying the inspection phase to the project or not. This model based on K- Mean's algorithm. Our model serve 5 classes. When a project passed the design phase according the requirement of the project the model will be applied. According to the result the company decide pursue the inspection phase or not. In our database we have analyzed the 50 project. We have divided each project into two phase like design and analysis phase. After that. We quantitate these phases. After that we applied clustering algorithms.

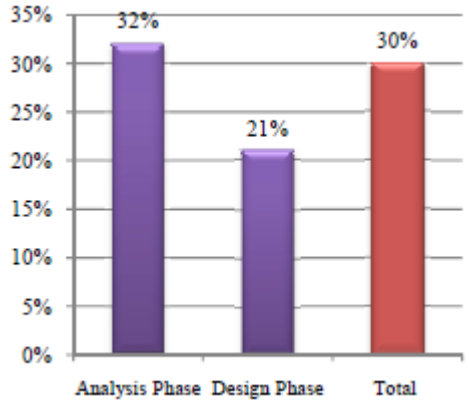

Figure 3. Development Phase

As result the proposed model help the project manager to apply the inspection phase to project or not. We notice that according to the result the project applied this model resulted with % 32 success.

Reference

[1] David L . Parnas , Senior Member, IEEE, and Mark Lawford , Member, IEEE, ”The Role of Inspection in Software Quality Assurance”.

[2] N.H TABA, S.H OW, “A Scenario Based Model to Improve the Quality of Software Inspection Process”, Department of Software Engineering, University of Malaya.

[3] Walter W. Schilling, Jr. Department of Electrical Engineering and Computer Science Milwaukee School of Engineering, “Teaching Software Inspection Effectiveness: An Active Learning Exercise”.

[4] S.M.DJ.T. Jayatilake, S.K.K.M. De Silva, U.T. Settinayake, S.A.S. Yapa, J.M.D.A.M.M.S Jayamanne, A.G.A.M. Ruwanthika and C.D. Manawadu , Department of Information Technology, Sri Lanka Institute of Information Technology (SUIT) Colombo, Sri Lanka. ” Role of Software Inspections in the Sri Lankan Software Development Industry”.

[5] Lionel C. Briand, Khaled El Emam, Bernd Freimut, Oliver Laitenberger,Fraunhofer Institute for Experimental Software Engineering. ” Quantitative Evaluation of Capture-Recapture Models to Control Software Inspections”.

[6] Trent Kroeger and Neil Davidson, Centre of Excellence in Defence and Industry Systems Capability Defence and Systems Institute, University of South Australia. ” A Perspective-based Model of Quality for Software Engineering Processes”.