Mechanism For Detection of Software Design Defects

KalalaliRoselineAsimini-Hart [#1], BennetOkoni[*2], Nuka Nwiabu[#3]

# Department of Computer Science,
Rivers State University, Port Harcourt,
Rivers State Nigeria

Abstract

This dissertation provides a mechanism for the detection of software design defect. There are stages of design defect which includes planning, analysis, design, implementation and testing in which this dissertation focuses on the design process alone. There are basic principles of software design which are ambiguity, inferiority, inconsistency and incorrectness. This dissertation is picking inconsistency and incorrectness as the two variables to work with to test5 for software defect. The design methodology is an object-oriented design which also has five stages to actualize the aim of this dissertation. The first stage of the OOD is been used which is to define the context and external interaction with the system thereby produces an SRS (Software requirement Specification) Document with the developer. After developing the software, the tester tests the software using the two chosen variables to test against the SRS Document to ascertain whether the software developed is in conformity with the laid down document. In the testing process, expert system is used. machine learning under the supervised learning where the system is trained with an algorithm and the SRS data stored into the database.

***Keywords** – software, design, OOD, software efficiency, software inconsistency,*

I. INTRODUCTION

Software is a set of instructions and associated documentation that enables a computer to perform a specific task as opposed to the physical components of the system (hardware) which comprises of system and application software.

A defect is a general word for any kind of shortcoming or imperfection, whether literal or figurative. Sometimes applications can malfunction and produce incorrect results, usually resulting from coding or logic error. In such an instance, we call it a software defect.

Several methods may be used to identify or detect defects at different phases of the development life cycle. It is well known that the later defects are detected the more expensive they are to fix or resolve. As such, organizations seek to identify potential defects as early as possible in the software development process. Early defect detection using static code analysis eliminates future costs and prevents problem expansion in the production or implementation phases. Preventing a problem will always be a better practice than waiting for it to surface on its own. The earlier a development team can identify flaws or vulnerabilities within the software, the less the issue will cost and the easier it will be to remedy. Early defect detection using static code analysis ensures that the end product has minimal-to-no defects, thus creating a stable infrastructure deployment or market release.

According to Capers Jones (20l2), the expense of fixing defects has been the most expensive in all software dealings. Given the fact that defect repairs are the most expensive element in the history of software, it might be expected that the cost is measured accurately and carefully. In avoidance to measure defect repair cost, lines of codes are used by some companies as metrics that are against normal economic assumptions that disqualify high-level language and ‘cost per defect’ which condemns quality.

A synergistic combination of formal inspections, static analysis, and formal testing can achieve combined defect removal efficiency levels of 99%.

Static code analysis is possible as soon as there is code being written. Early defect detection assessments provide a cost-effective, accurate approach to creating a secure, robust software. As flaws or vulnerabilities remain unidentified, they create a continuously compiling expense in the form of cost-to-fix or remediation. Automated software using static analysis offers direct insight as to the source of the problem and makes it easy to resolve issues long before deployment. This drastically decreases organizational costs, increases developer productivity, and ensures a reliable implementation or release.

CAST Application Intelligence Platform or AIP is an enterprise-driven static and architectural analysis solution that assesses size, complexity, quality, and technical risk at any point in the development life cycle.

II. MATERIALS AND METHODS

Research methodology is the study of how a specific research project is carried out using some laid down techniques or approach. To effectively obtain the objectives of this dissertation, A Constructive research method is used to structure the research goals. The design approaches are Top-Down design approach and object-oriented Design Approach (OODA) for developing the system configuration.

A. Constructive Research Method

Constructive research is one of the very common computer science research procedures. The constructive approach means problem solving through the construction or use of models, diagrams, plans, organizations, etc. This style of study is generally used in technical sciences, operations analysis, mathematics, clinical medicine and in operations research (Eero et al, 1993). The word "construct" is frequently used in this context to indicate a new contribution being developed and the construct can be a new theory, model, software, algorithm, or a framework. Mathematical algorithms and new mathematical entities present theoretical examples of constructions. Constructive method solves practical problems while producing an academically valued theoretical contribution (Liisa et al., 2017). The main idea of the Constructive study is that construction, based on the existing or present knowledge is used in a fresh or new way, with possibly adding a few missing links (Crnkovic, 2010).

Constructive research help in solving practical problems while producing an academically appreciated theoretical contribution. The solutions, that is, constructs, can be processes, practices, tools or organization charts for improved apprehension to the reader as well as the researcher.

Relevance of constructive research to the research work are:

I. Help to obtain a comprehensive understanding of the study area;

II. Creating one or more applicable solutions to the problem;

III. Demonstrating the solution's feasibility;

IV. Connecting the results back to the theory and demonstrating their practical contribution.

The constructive method is illustrated by dividing the study into the stages:

I. Discover a practical relevant problem which also has a good research potential.

II. Acquire general and comprehensive understanding of the topic.

III. Create (constructing a solution).

IV. Show that the solution works.

V. Show the theoretical relations and the research contribution of the solution concept.

VI. Examine the scope of the solution.

B. Problem / Solution

This is the process or act of finding the mechanism for detection of software design defects solution using a particular tool. Constructive research method help to place the problem into a particular context that spells out the parameters of inquiry.

A defect in a software design always creates a recurring expense in the form of cost-to-fix or remediation. This can be temporarily fixed by creating a virtual operating system on which the application software is to be tested upon for defects.

C. Practical Relevance

This involves having a practical understanding of the mechanism of detecting a software design defect solution, which means that they help us to be fully involved in the work. However, in some disciplines, it may be more important that a dissertation has practical relevance. Research that has practical relevance adds value; for instance, it could make a recommendation for a particular industry or suggest ways to improve certain processes within an organization as being more concerned with or relevant to practice than theory.

D. Theoretical Relevance

Theoretical relevance is formulated to explain, predict, and understand phenomena and, in many cases, to challenge and extend existing knowledge within the limits of critical bounding assumptions. The theoretical framework is the structure that can hold or support a theory of mechanism for the detection of software design defects solution gives the new theoretical knowledge that needs scientific.

E. Theoretical body of knowledge

This is the complete set of concepts, activities and terms which comprise a professional domain, recognized by the relevant professional association. It is a type of knowledge representation by any knowledge organization.

An information source is a person, thing, or place from which information of the causes of software design defect comes, arises, or is obtained. Information sources can be known as primary or secondary. That source might then inform a person about something or provide knowledge about it.

Constructive research method in project management research was performed by Oyegoke (2oll), which support the application of the constructive research method to the construction of Project Management discipline. In (Lisa et al, (2ol7), Problem Solving for Complex Projects was researched using the Constructive Research method.

F. Methods of Data Collection and Tools

The method of data collection and tool adopted in this research is the secondary data collection method which is further classified into Internal and external secondary data. They include;

I. Information collected through censuses

II. Internet searches or libraries

III. Progress reports

The rationale for selecting secondary data collection in this thesis is that it is time-saving, cost-effective, available from other sources and may already have been used in previous research, making it easier to carry out further research and much of the background work needed for the mechanism for detection of software design defects have already been carried out.

G. Design Methodology

Design methodology refers to the development of a system or method for a unique situation. Today, the term is most often applied to technological fields in reference to web design, software or system architecture. The main thing in design methodology is designing unique solutions tailor-made for each unique situation, be it industrial design, architecture or technology. The methods used in this research is “Object-Oriented Method and Recursive Design” (OOM/RD).

H. Object-oriented Method

The software development methodology uses object-oriented development methods for which the entire system is broken down into subsystem and modules and the modules are seen as objects. Object-oriented programming is often the most natural and pragmatic approach, once you get the hang of it. An object oriented programming language allows you to break down your software into bite-sized problems that you then can solve, one object after the other. The steps in a typical object-oriented Method **(**OOM) model will generally include the following details:

i. Identification of critical objects of the main systems design by breaking them down into modules (smaller blocks) or subsystems

ii. Performing software processing on identified objects

iii. Re-applying software processing on the identified objects

The steps are crucial to almost any object-oriented technological design and must performed in a repeating manner to arrive at reasonable performance estimates. Bottom-up design is the type of object oriented method used in the system design because it is the best for systems that are created from the current system. This method is chosen for this research since the modules in the proposed method is seen as objects.

I.Recursive Design

The research utilized the Recursive design approach because it affords programmers the advantage of the repetitive structure present in many problems, especially in algorithm design.

However, the main reason of combining object oriented method and recursive design in this design methodology of this research is for a good userinterface of the system design making the system user-friendly.

Using the “Recursive Design” method in this research makes me to understand the domain of “big data” in the ranking method which will be used in solving the issue because of it ability of Iteration. This is as result of multi-test of the process to generate a sequence outcome.

J. System Architecture

An architecture specification is a precise description and illustration of the system, which is constructed in a pattern that provides insight into the structures and actions of the system. Figure 3.l.Shows the proposed system architecture

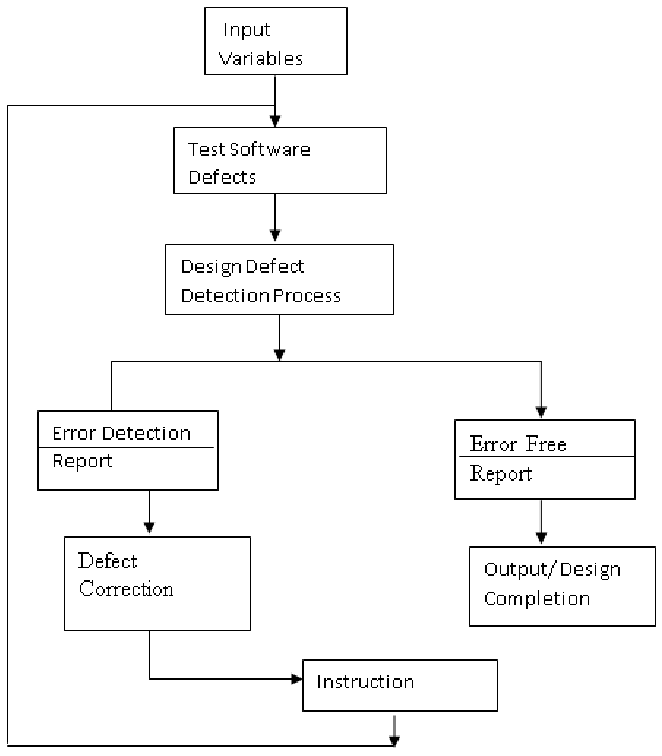

Figure 1 System Architectural Structure

The principles of software design are incorrectness, inconsistency, ambiguity, and inferiority. For this research, the scope in narrowed down to two principles/parameters which are; inconsistency and incorrectness.

i. Incorrectness: The design does not meet the user's requirements on its functionality and features. Such an error may appear in the form of misinterpretation or omission of user's requirement

ii. Inconsistency: This is where a design fails to work. For example, if two design statements make conflicting assumptions about the functionality of a component or the meaning of a data item, the design simply does not work.

K. Design Defect Detection Process

This process is of two levels Primary and secondary. The primary defect detection process is at the same level of abstraction and the secondary defect detection process is at a different level. It is noted that design defect, design inspection, and functional testing are primary processes code inspection and unit testing are secondary defect detection processes

L. Report

In this stage, the process reports the result of the defect and the required changes to be done in the design to achieve the specific requirement earlier specified.

M. Detailed Design of Functional Testing

The detailed design explain the basic component they make up the operation of the system. Figure 2 below shows the detailed design of the functional testing system and the component. The process of arranging the preliminary design of a system or component in such a manner that the design is sufficiently complete to be implemented.

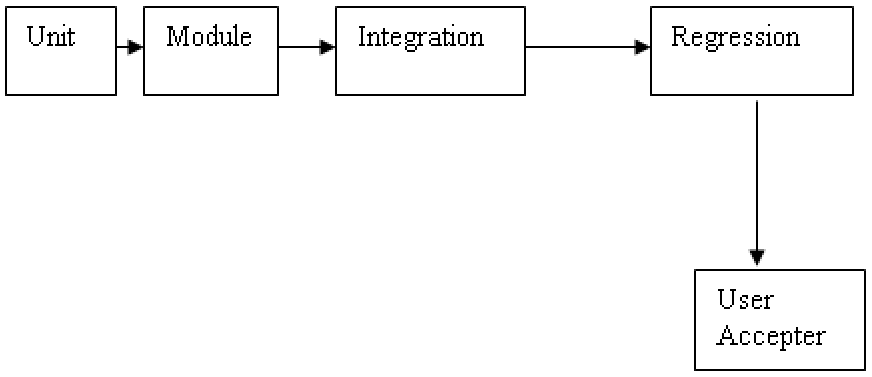

Fig. 2: Component of Functional Testing

In the proposed system, a Mechanism for the detection of software design defects is proposed. In the case of software defect prediction models, there is four-parameter use in predicting whether the software is defective or clean.

Algorithm 1:

```
// return the greatest
// common denominator
Int Euclid (int m, int n)
 {
        Assert {n > m};
        Assert {m>0}
        Int r;
        If {n > m}
         {
        r = m;
        m = n;
        n = r;
          }
        r = m     % n;
        while {r ! - 0}
          {
        m = n;
        n = r;r = m % n;
          }
        assert (n > o);
        return n;
}
```

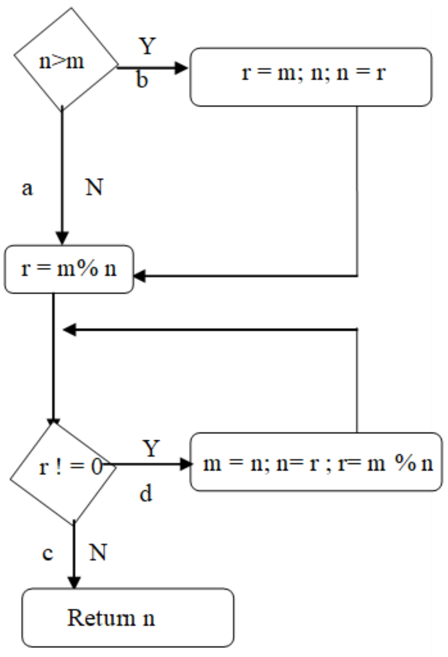

Fig. 3: Testing flowchart

The following figure below shows how integration testing can be done without performing fill basis path testing of all modules.

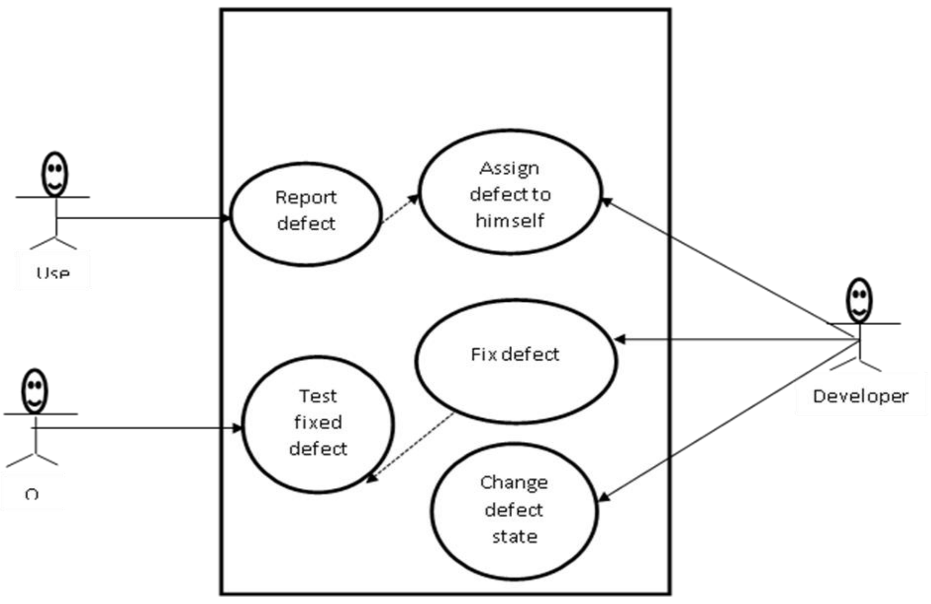

Fig4: Use case diagram of the proposed system

A use case is a process of modeling a system's function in terms of system events and how the system responds to those events. In Use Case modeling, Use Case Diagram contains symbols written in Unified Modeling Language (UML).

The system functions include in this case are the user responsible for reporting defects while the quality access (QA) is responsible for testing and fixing defects and the developer is will handle the change of defect, fix the defect and also assign defect to himself in the cause of developing.

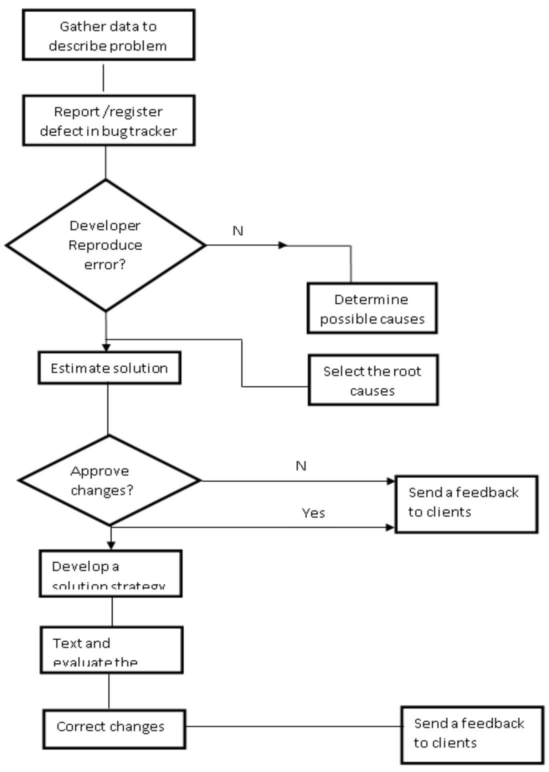
Gather data to describe problem
Report /register defect in bugtracker
Developer Reproduce error?
N
Determine possible causes
Estimate solution
Select the root causes
Approve changes?
N
Yes
Send a feedback to clients
Develop a solution strategy
Text and evaluate the
Correct changes
Send a feedback to clients

Fig. 6: Activity Diagram

The activity diagrams describe the various activities that are involved in this research. It is laid down activity that the system follows to achieve its aim. The activities of this work start from gathering of information to describe the software design defect, after the gathering of data it will report or register defect in bug tracker if the developer cannot reduce the error, he will determine the possible cause and select the root cause then estimate solution if the approved change is no it will send feedback to clients else it will develop a solution strategy maybe test and evaluate the solution and correct changes.

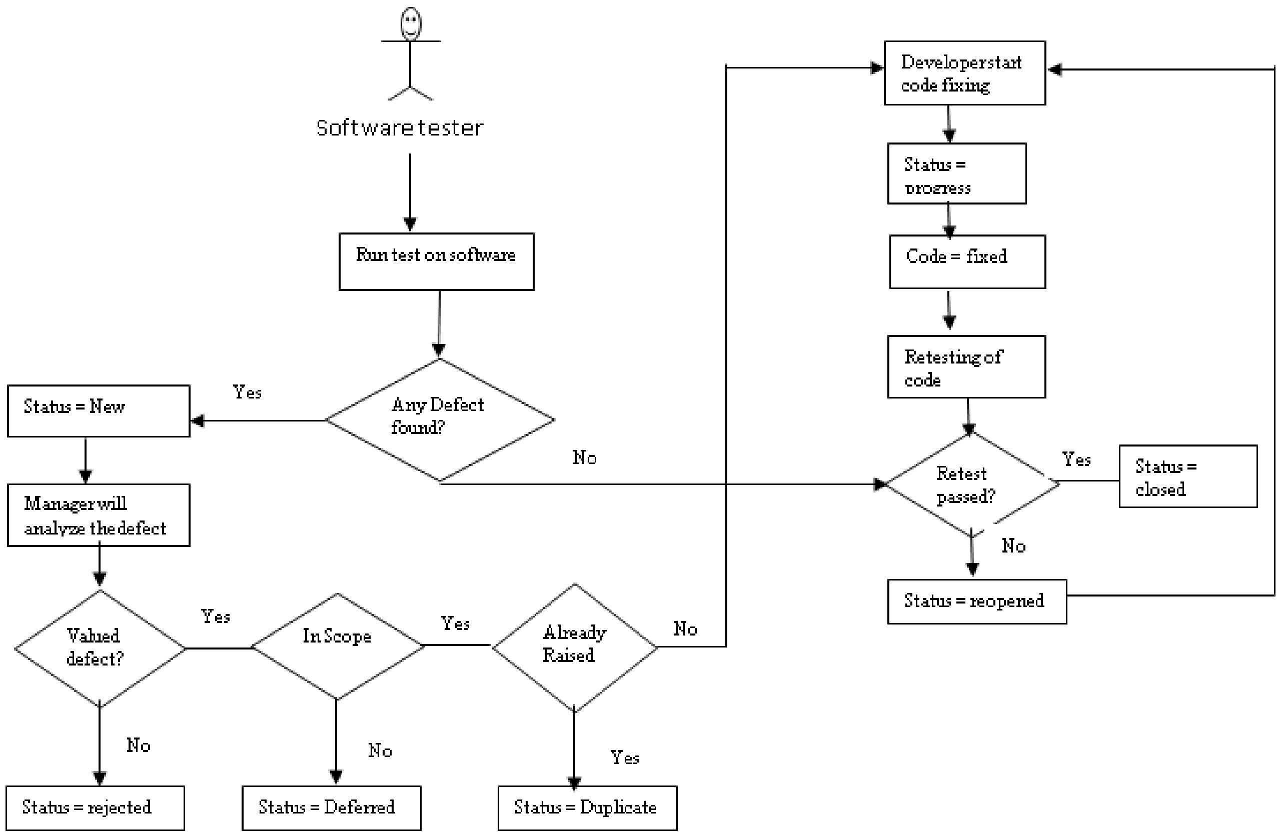

Fig. 7: Flow diagram of the system mechanism

Figure 3.7 shows the mechanisms and the processes that take place in software design defect they include:

Analysis: in analyzing a software defect, a tester must be an intelligent observer and should likewise have a lot of knowledge in such a manner. A tester is required to apply demonstrative perception to have the capacity to reveal the test situation from both a positive and negative approach

New: when the defect is reported and posted for the first time. Then it is given a new state.

Duplicate: if the bug was reported earlier and again it is reported then the status is marked as duplicate.

Rejected: If the bug reported the testing team is considered as working fire then the defect is rejected by the development form.

Deferred: the defect reported by the testing team is not of high priority and can be fixed in next releases then the defect is marked as deferred and will be fixed in an upcoming release

Assigned: once reported the lead of the testing team will approve the defect is genuine and the defect is assigned to the corresponding developer. Then the status is changed to assign.

Open: once the defect is assigned to a developer, the status of the defect is changed to an open state.

Fixed: the status of defect changed to fixed when the developer makes the necessary changes in code and verified the changes.

Retest: after the defect is fixed by the developer, it is assigned to the testing team for a retest. The status is changed to retest.

Reopen: once the testing team retests the fixed defect. If the defect is still present, there then the status of defect is changed to reopen.

Close: if the testing team retests

N. The architecture of Ambiguity Detector
The architecture of the ambiguity detector is a tool that contains four components i.e. SRS document, Algorithm for detecting sentence, Corpus of different ambiguous words and parts of speech.

O. Software Requirement Specification (SRS)Document
SRS (Software Requirement Specification): is the primary vehicle for agreement between the developer and customer on exactly what is to be built. It is a document reviewed by the customer or his representative and often is the basis for judging the fulfillment of contractual obligations.

The SRS records the result of the problem analysis. Documenting the result of analysis allows questions about the problem to be answered once during development. It defines what properties the system must have and constraints on its design and implementation. It helps in ensuring that requirement decision is made explicitly during the requirement phase not implicitly during programming.

P. Unit Testing of Software
Unit testing is a software testing method in which the individual units of the source code such as the associated functions are tested to determine whether each unit of the code generates the precisely expected output. In this, the unit test cases for every microprocessor machine instruction set were developed and used to verify the incorrectness and inconsistency of the instruction set operations emulated in the tested.

III. RESULTS AND CONCLSION
This section explores facets of the application under review and evaluates its implementation and performance with regards to its purpose for which it was designed. It showcases the design specification, installation procedures, and documentation and launching.

Our application compilation for software defect mechanism is grasped using PHP (HyperText Preprocessor) as the main scripting language, CSS (Cascading Style Sheet) was used to style the interface, MYSQL Server as the database server, and XAMPP as the Web server.

A. Result for incorrectness
the resulting level is determined by the threshold (th) level
th< 25 represent Error Free and
th>25 represent Error

B. Result for inconsistency
the resulting level is determined by the threshold (th) level
th<50 represent Error Free and
th>50 represent Error

TABLE I
Application name: Like me

Variables	True (<25) False (>25)	Point Per Variable %	Solution Preferred
Incorrectness	False	83	Make sure all finding given by the owners are met
Inconsistency	✓	N/A	N/A
Total Amount of Defect		83	

TABLE II
Application Name: Global Development

Variables	True (<25) False (>25)	Point Per Variable %	Solution Preferred
Incorrectness	False	48	Make sure all finding given by the owners are met
Inconsistency	N/A	N/A	N/A
Total Amount of Defect		48	

TABLE III
Application Name: Social Network

Variables	True (<25) False (>25)	Point Per Variable %	Solution Preferred
Incorrectness	False	52	Make sure all finding given by the owners are met
Inconsistency	N/A	N/A	N/A
Total Amount of Defect		52	

TABLE IV
Application Name: Smart Home Design

Variables	True (<25) False (>25)	Point Per Variable %	Solution Preferred
Incorrectness	True	N/A	Good condition
Inconsistency	✓	64	N/A
Total Amount of Defect		64	

A. Graphical Representation of the Result

The graphs below represent the incorrectness of software design defect as a variable, it is assumed that when the threshold is greater than 25% it will be abnormal this is represented by the red symbols while the normal is less than 25%. This applies to the inconsistency variable but in the ambiguity, the threshold is set at normal when is less than 50% while the abnormal is greater than 50%.

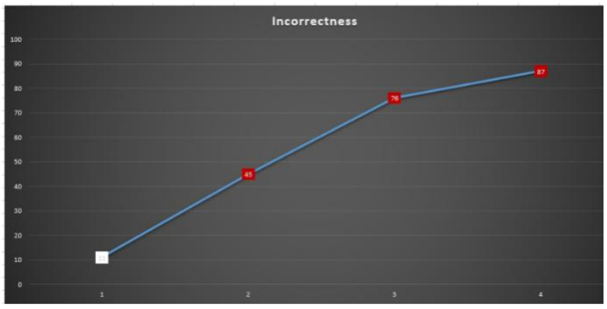

Fig. 8: incorrectness of software design

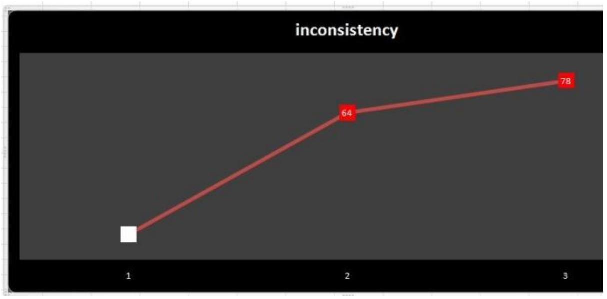

Fig. 9: inconsistency variable

This studywas narrowed down to software design defect which is the art of testing software based on an input variable that determines an error-free software or defect software. Utilizing static code analysis to detect defects helps averts future costs and prevents problem expansion in the production or implementation phases. Essentially, faults detections activities are carried out in every phase of the software development life cycle based on their need and criticality. In this application, we collect the information of the said software and analyze their must variables which are the functionality, reliability, usability, efficiency, maintainability, and portability, when we look into these features of software design decisions are made. So, this mechanism works when we put in a variable and we check for the software design defect the system will analyze the input and give the required result if the software is designed defected or error-free.

IV. CONCLUSION

The detection and correction of design defects due to poor design choices are difficult because of the lack of precise specifications of defects and tools for their detection and correction. No single software fault detection technique is capable of addressing all concerns in error detection. Similar software reviews and testing, static analysis tools (or automated static analysis) can be used to remove faults before a software product release. Inspection, prototyping, testing, and proofs of correctness are several approaches to identify faults. in software defect solution testing is one of the least effective techniques. A single error can lead to one or more faults and several faults can lead to failure. To avoid this failure in software products, faults detections activities are carried out in every phase of the software development life cycle based on their need and criticality.

To this end, it must be emphasized that if all parties both middle level and top management, company owners and all stakeholders in software market are committed and agree to define and meet the expectations of each other by producing error-free software our aim of maintaining error-free software will be achieved since success is measured by achieving defined goals and meeting expectations to the satisfaction of all.

REFERENCE

[1] Dresch, Aline, Lacerda, Daniel Pacheco; Jr. Ose Antonio Valle Antunes (2015). "*Design Science Research: A Method for Science and Technology Advancement Cham*". Springer, pp. I doi:10, 1007/978-3-319-07374-3.
[2] Hoshggoftaar, T., & Allen, E. (1998). "*Predicting the order of fault-prone Modules in legacy software*".
[3] Kemerer, C. F. & Mark, C. (2009). The impact of design and code reviews on software Quality: "*An Empirical study based on PSP Data*" IEEE Transactions on Software Engineering. Vol. 35 No. 4.
[4] Liu, Khoshgoftaar, &Seliya (2010). "*International Conference or Machine Learning and Applications*".
[5] Zhang Yanjun, "*Design and Development of Chinese Teaching Software based on Chinese Audio-visual Multimedia Corpus*", SRG International Journal of

Electrical and Electronics Engineering Volume 5 Issue 10 Oct 2018

[6] Mende, T., &Koschri, R. (2009). "*Revisiting the evaluation of defect prediction models? Process International Conference on Predictor models in software engineering*".

[7] Ohlsson, N., &Alberg, H. (1996). "*Predicting fault probe software module in telephone switches*" IEEE Trans Software Engineering, 22(12), 886-894.

[8] OMG (2010). "*Unified Modelling Languages, superstructure specification*, version 2.1.1, http://www.omg.org/spec/UML/2.1.1/superstructure /PDF/.

[9] Orthogonal defect classification, "*A concept for In-Process Measurements, IEEE Transactions on Software Engineering*", SE 18. P. 943 – 956.

[10] Wise, A. (2006). Litter-JIL. "*1.5 Language Report on Technical Report*", Department of Computer science, University of Massachusetts.

[11] Zheng, J., Williams, L., Nagappan, N., Snipes, W. Hudepohl, J. P. &Vonk, M. A. 2006). "*On the value of static Analysis for fault Detection on Software*". IEEE. Transactions on Software Engineering, 32, (4).