JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

AN OVER VIEW OF STATIC TEST TECHNIQUES IN SOFTWARE DEVELOPMENT PROCESS

[1]Dr. K.Sai Prasad Reddy, [2]Prof. K.Nagabhushan Raju
[1]Mentor, [2]Head of the Department
[1]Center for Skill Development, Entrepreneurship & Incubation,
[1]Sri Krishnadevaraya University, Ananthapuramu, India

Abstract: Static testing is a form of software testing. Static test techniques plays key role in Software development process. These techniques provide an authoritative way to improve the quality and productivity of software development. Static testing techniques are used to check the sanity of the code, Algorithms and all project documents. The fundamental objective of static testing is to improve the quality of software work products by assisting engineers to recognize and fix their own defects early in the software development process. Static techniques can improve both quality and productivity by impressive factors. Effective static test techniques reduces the development time scale and also reduces the testing time and cost. Static testing will be performed by the Software development team who writes the code and by Software testing team who involves in testing activities.

IndexTerms **- Inspections, Reviews, Static Analysis, Walkthroughs.**

I. INTRODUCTION

Techniques used for static testing are known as Static Testing Techniques. The static testing techniques depend on the manual assessment and automated analysis of the project code and project documentation. Static testing is a form of testing where project/application to be tested is not executed. Instead project/application code is checked for conformance to the functional requirements, design, missed functionality and coding errors. Errors which are found during static testing are faster and cheap to resolve than the defects found during dynamic testing phase or even at later stages. During static testing code will not get executed. During static testing typical defects like deviations from client standards, client requirement defects, project design defects, insufficient maintainability, incorrect interface specifications etc,. will be identified. Static test techniques are implemented from project initiation or early stage of the Software Development Life Cycle i.e. from the requirements stage. This will help in finding the defects at the early stage of the software project. By finding the defects at the early stage we can avoid defects multiplication. Static testing techniques will be implemented at all stages or mile stones of Software development Life Cycle, Static Testing Techniques are mainly classified as 1. People based techniques 2. Tool based techniques.

Benefits of static testing techniques are:

- Development productivity improvement
- Reduced development timescales
- Testing time and cost can be reduced
- Lifetime cost reductions
- Reduced fault levels
- Improved customer relations

II. PEOPLE BASED TECHNIQUES

A. Reviews
B. Walkthrough
C. Inspection

A. Reviews

A group of persons or team looks for errors, mistaken assumptions, lack of clarity and deviation from standards or client requirements. Review is an activity carried out to verify the code and documents at different stages for its completeness, correctness & consistency [1]-[6]. This verification is done w.r.t to client requirements or previous documents in the Software Development Life Cycle or with respect to established standards or norms that have been agreed upon. A review is performed as a manual activity, but there is also tool support. Review is a static testing which will be performed on software work products, project code and has to be performed well before dynamic testing. Goal of Review is to identify defects within the stage or phase of the software project where

they originate, rather than in the later stages. This is referred to as "Phase containment". Reviews commonly find errors that are not possible to detect by regular testing [2]. Reviews also provide a form of training, including technical and standards related to every participant. From testing point of view, we can use reviews to allow ourselves to be involved much earlier in development lifecycle. As at the beginning of the project there is nothing you can physically test, you can involve in the review process of various documents. From team involvement at this very early stage of the development lifecycle, team will have a fair idea of requirements and testable items. This will gives to team a head-start on thinking about how team has to approach testing the requirements also. During the review process team will find defects like missings, gaps, corrections, clarifications etc,.

Distribution of Defects in Software Development Life Cycle

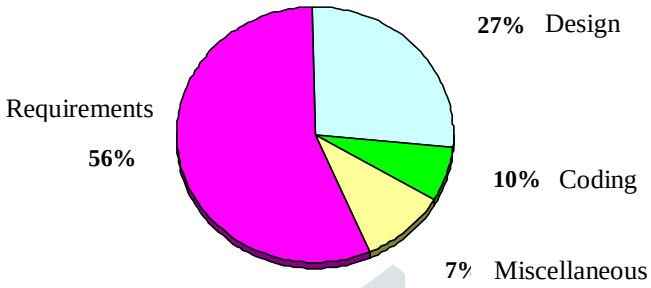


Fig.1 Distribution of Defects in Software Development Life Cycle

Steps in Reviewing of Project document

- Reading each and every line of the document
- Analyzing each line of the document
- Understanding each line of the document
- Finding and reporting the defects to the concerned author of the document or code.

The typical format of defect log document for reporting the defects which are found in reviewing of project document. This defect log is provided to record the defects like Missings, Gaps, corrections, clarifications and suggestions provided by reviewers of each document during reviews. Defects regarding individual typographical, grammatical and spelling corrections will be also reported in this defect log document. All defects regarding content e.g. accuracy, consistency, etc,.
or pervasive quality concerns can be recorded in this defect log document.

Table 1 Defect log document.

Project Id:			Project Name:			Document Name:	
Defect ID.	Name of the Reviewer	Date of Review	Defect Type	Page No.	Defect Description	Resolution	Remarks

General classification of Reviews

Informal Review
Generally a one to one meeting between the author of a work product and the peer, initiated as a request of input regarding a particular artifact. Informal reviews Costs low and are widely used. These reviews have no formal process [3] . Informal reviews can be conducted for project documents, designs and code and result may be documented. Pair programming is an informal review method.

Formal Review
Facilitated by a knowledgeable individual called a moderator, who is not a member of the team or the author of the product under review. Full participation of all the members of the review team is required. The issues raised are captured and published in a formal report distributed to participants and Management .

Types of Reviews

In-process Review: To verify the work product during a specific time/period of a life cycle to identify defects as works progresses. Ex: In between any process like software requirements, detailed design etc.
Phase-end Review: To verify and to determine whether to continue with the planned activities. Ex: At the end of software requirements, detailed design, test readiness, etc.
Post-implementation Review: Done after the implementation is completed. Process based on actual results and to identify the opportunities for improvement. Also called as postmortems.

Stages of Formal Review Process [4]

Planning: There should be awareness of company policies, product requirements and project plans that may provide specific requirements in order to perform review. Personnel selection and Entry criteria & Exit criteria should be defined at this stage.
Kick-off: Moderator ensures that the item(Document/Design/Code…) is ready for review and will be distributed to all participants. Moderator also briefs the attendees on their roles and responsibilities.
Preparation: All review participants examine the item to be reviewed. Reviewers checks for deviation from the standards. Comparison to similar items can also be used.
Meeting: The review normally continues for One to Two hours. All items on the agenda/checklist are worked through. Findings are noted by the recorder. Comments are aimed at the review item and not at the author.
Re-Work: The Moderator and Scribe document the findings in a Review Summary and report it to the Manager. The Review Summary contains the defects found and actions of follow-up work to be carried out.
Follow-up: The Moderator ensures that all additional work by the author is checked for completeness and correctness. An additional review may be required; dependant on the amount/complexity of re-works undertaken.
Exit Criteria: It can take the form of ensuring that all actions are completed, or that any uncorrected items are properly documented, possibly in a defect tracking system.

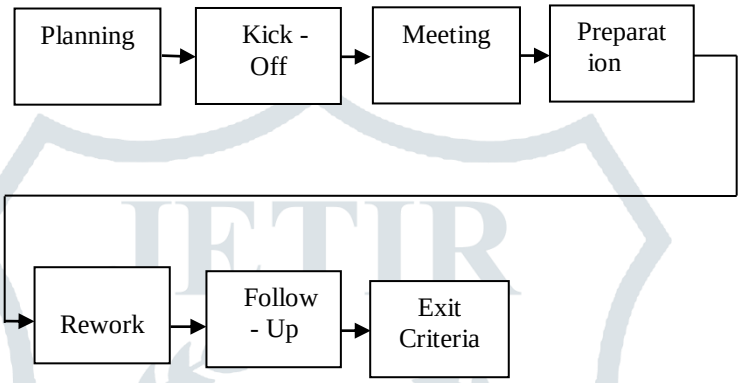


Fig.2 Formal Review Process

Roles and Responsibilities in Review Process

There are different roles and corresponding responsibilities in review process.
Manager: The Manager is the person who makes the decision to hold the review. Managing people's time with respect to the review is also one of the responsibilities of a Manager. Moderator: The Moderator has overall control and responsibility of the review. He/she schedules the review, controls the review and ensures any actions from the review are carried out successfully. Moderator will be provided training to execute his role successfully.. Author: The Author is the person responsible for creating the items to be reviewed. The Author may also be asked questions during the review. Reviewer: The reviewers are the attendees of the review and responsible for finding errors in the item under review. In order to provide a well balanced review of the item, reviewers should come from different perspectives.Scribe: The Scribe or Recorder is the person who is responsible for documenting the issues raised during the process of the review meeting

Importance and Advantages of Review

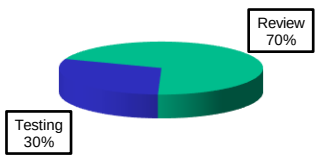


Fig. 3 Defect Identification efficiency

- 10 times reduction in faults reaching test, testing cost reduced by 50% to 80%
- Reduce faults by a factor of 10
- 25% reduction in schedules, remove 80% - 95% of faults at each stage, 28 times reduction in maintenance cost, many others
- Reviews are almost 70 percent efficient in finding defect when compare to testing.
- As defects are identified by the review process in the earlier part of life cycle, they are less expensive to correct.
- Reviews are an efficient method of educating a large number of people on a specific product/project in a relatively short period of time.

Methods of Review [5]

1. Flow Chart: The program is explained from a flowchart of the program logic.
2. Source Code: The review looks at each line of code to understand the program.
3. Sample Transactions: The lead programmer explains the programs by explaining the processing which arise representative sample of transactions.
4. Program Specifications: The program specifications are reviewed in the matter of understanding the program.

B. Walk-through

Walk through is an Informal activity and will be conducted by the concerned author in which members of the group are also involved [3]. It's an informal verification process by which some inputs given, that may or may not be considered for any changes. Walk-through is used to aid Learning. This is basically carried for obtaining a second person view on the activity carried out. Basically walk–through is done for the source codes e.g., Improving Programming Logic, Design approach etc

Sample criteria for Code walk-through
Minimize or eliminate use of global variables.
Use descriptive function and method names - use both upper and lower case, avoid abbreviations
Organize code for readability.
Use white space generously - vertically and horizontally
Each line of code should contain 80 characters max.
One code statement per line. etc.

C. Inspection

A formal assessment of a work product conducted by one or more reviewers to detect defects, violation of development standards, etc,. During inspection process defects are identified but inspection process do not attempt to resolve them[2]. During every stage of project development lifecycle, anything that is written can be Inspected . Inspection can find deep-seated faults. All deep-seated faults can be corrected.

What can be inspected?

- Policies,
- Strategy documents
- Business Plans
- Marketing or Advertising Material
- Contracts
- System requirements
- Feasibility studies
- Acceptance Test Plans
- Test Plans
- Test Designs
- Test Cases
- Test Results
- System Designs
- Software Code
- User manuals
- Procedures
- Training material etc,.

What can fail the Inspection Process?

- No appreciation for the fundamental importance of Rules.
- Slow checking rates.
- Not following the strict entry and exit criteria.
- False logging rates.
- Amount of responsibility given to the author

Inspection will be better and worthwhile only if the following criteria are satisfied [2]

- Entry criteria for inspection:
- Trained resources
- Optimum checking rate
- Prioritizing the words
- Following the standards
- Focus on constant process improvement
- Definite exit criteria

- Quantified estimates of remaining major faults per page

The following diagram illustrates the typical Inspection process

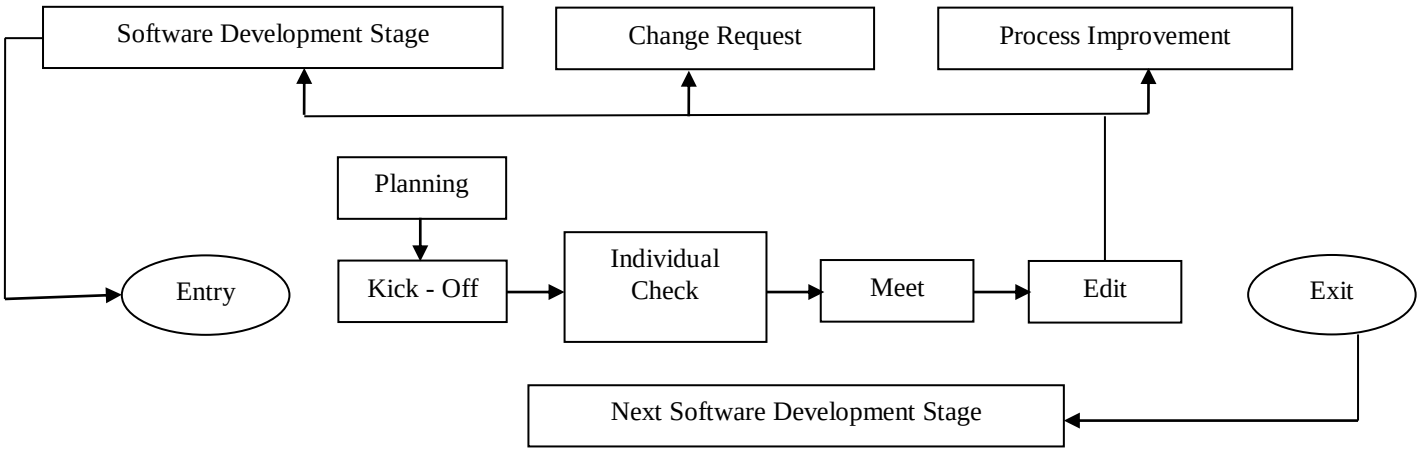


Fig. 4 Typical Inspection process

The inspection should demonstrate the effectiveness

Table 2. Effectiveness of Inspection

	Effectiveness	Return on Investment
Typical Review	10 – 20 %	Unknown
Early Inspection	30 – 40 %	6-8 Hrs/Inspection hrs
Matured Inspection	80 – 95%	8 – 30 Hrs/Inspection hrs

III. STATIC ANALYSIS

Static analysis is a set of methods designed to analyze software code without executing the program. Static Analysis provides an understanding of structure of the code which ensures that whether code is as per industry standards or not. So, by using Static analysis, we can effectively test the program even before it is written [7].
Some advantages of using Static Analysis are: Finding defects before any tests are even run early warning of unsatisfactory code design finding dependency issues, such as bad links etc.

Static Analysis will detect the following types of defects

- Unreachable code
- Uncalled functions
- Undeclared variables
- Programming standard violations
- Syntax errors

Static Analysis is commonly performed by automatic processes. In the process of Static Analysis, code is checked for violations of standards and for any faults. For example, A „Compiler" performs Static Analysis when it detects problems. So, in the process the compiler statically analyses code, and "knows" a lot about it by finding variable usage, syntax faults, unreachable code, undeclared variables, parameter type mismatches, uncalled functions and procedures, array bound violations, etc [8].

Static Analysis Methods

A. Data Flow Analysis
B. Control Flow Analysis
C. Cyclomatic Complexity Measurement

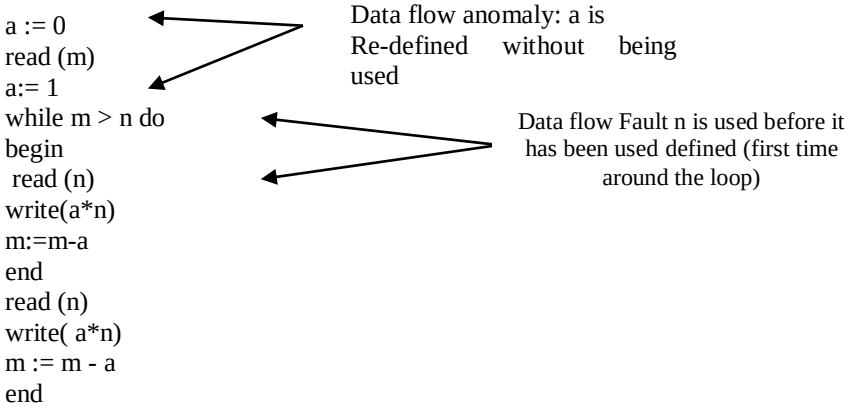
A. Data Flow Analysis

It is the process of collecting information about the way the variables are defined and used in the program. Analysis is done at basic block granularity. Collected information is represented as a set of data flow equations useful for performing several optimizations, such as, constant propagation and copy propagation [9].
Look at the following code to understand the difference between the concept of variables used and variables defined.

```
x = y + z
IF a > b THEN read(S)
```

In the first line of code x is defined, whereas y and z are used. If you can see the second line a and b are used and S is defined.

Data Flow Analysis Faults:



B. Control Flow Analysis

Control Flow Analysis displays the logic structure of software. The flow of logic through the program is charted. Control Flow Analysis is generally used only by Software Development team as it is low level testing and regularly implemented in structural Testing/Unit testing/Component Testing. This is used to conclude number of test cases that are required to test the programs logic. It can also provide confidence that the detail of the logic in the code has been checked [8].

Control Flow Analysis helps in analyzing and identifying:

- Nodes not accessible from start node
- Infinite loops
- Multiple entries to loops,
- Whether code is well structured, i.e. Reducible
- Whether code conforms to a flowchart grammar
- Any jumps to undefined labels
- Any labels not jumped to
- Cyclomatic complexity

The following is the example of unreachable code

```
Buffsize: 1000
Mailboxmax: 1000
IF Buffsize < Mailboxmax THEN
Error-Exit
ENDIF
```

In this case, Static Analysis finds the THEN clause unreachable, so it will flag a fault.

C. Cyclomatic complexity

Cyclomatic Complexity is the software metric that is used to measure the complexity of a software program. If we know how complex the program is, then we know how easy it will be to test. The complexity is calculated from a graph which describes the control flow of the software program. The formula for calculating the Cyclomatic Complexity from the flow chart is

$$C = \text{Number of Decisions} + 1 \quad (1)$$

In the shown flow chart number of decisions is Two. So the Cyclomatic Complexity for the shown flow chart is 3.

If the control flow graph is being used for analysis, then the Cyclomatic Complexity is calculated using the following formula

$$C = E - N + P \quad (2)$$

Where:

C = Cyclomatic Complexity
E = Number of edges
N = Number of nodes
P = Number of components

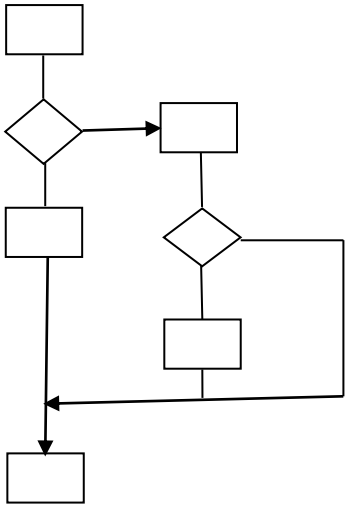


Fig. 5 Flow chart

Limitations of Static Analysis

- Cannot distinguish "fail-safe" code from programming faults or anomalies (often creates overload of spurious error messages).
- Does not execute the code, so not related to operating conditions.

Advantages of Static Analysis

- Can find faults difficult to see.
- Gives objective quality assessment of code.

For static analysis numerous tools are available and the majority of tools focus on software code. Static analysis tools are used by developers before and during component and integration testing and designers uses during software modeling.

What can static analysis do?

- It is a form of automated testing.
- check for violations of standards
- check for things which may be a fault

Static Analysis tool is descending from compiler technology. Compiler is best example for Static Analysis tool.

- A compiler statically analyses code, and "knows" a lot about it, e.g. variable usage; finds syntax faults
- Can find unreachable code, undeclared variables, parameter type mis-matches, uncalled functions & array bound violations, etc.

IV. CONCLUSION

Static Test techniques are more efficient and effective in finding defects at different stages in Software development life cycle. Defects that were found during static testing are much cheaper to remove. These techniques will prevent defect multiplication by identifying the defects within the stage or phase of the software project where they originate. The main advantages of static test techniques over dynamic testing are low cost of defects detection, enhancement of code and documents quality, identification of improvement opportunities.

REFERENCES

[1] Marc Roper, Murray Wood, James Miller, " An empirical evaluation of defect detection techniques, Information and Software Technology", Volume 39, Issue 11,1997,Pages 763-775,ISSN 0950-5849, https://doi.org/10.1016/S0950-5849(97)00028-1.

[2] A. A. Porter and P. M. Johnson, "Assessing software review meetings: results of a comparative analysis of two experimental studies," in IEEE Transactions on Software Engineering, vol. 23, no. 3, pp. 129-145, March 1997, doi: 10.1109/32.585501.

[3] V. R. Basili and R. W. Selby, "Comparing the Effectiveness of Software Testing Strategies," in IEEE Transactions on Software Engineering, vol. SE-13, no. 12, pp. 1278-1296, Dec. 1987, doi: 10.1109/TSE.1987.232881.

[4] Lott, Christopher & Rombach, Dieter. (1997). Repeatable Software Engineering Experiments for Comparing Defect- Detection Techniques. Empirical Software Engineering. 1. 10.1007/BF00127447.

[5] Sheikh Umar Farooq, SMK Quadri, "Empirical Evaluation of Software Testing Techniques – Need, Issues and Mitigation", Software Engineering : An International Journal (SEIJ), Vol. 3, No. 1, april 2013

[6] L. Dong et al., "Survey on Pains and Best Practices of Code Review," 2021 28th Asia-Pacific Software Engineering Conference (APSEC), 2021, pp. 482-491, doi: 10.1109/APSEC53868.2021.00055.

[7] R. Haas, R. Niedermayr, T. Röhm and S. Apel, "Recommending Unnecessary Source Code Based on Static Analysis," 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE- Companion), 2019, pp. 274-275, doi: 10.1109/ICSE-Companion.2019.00111.

[8] Wang Wei, Meng Yunxiu, Han Lilong and Bai He, "From source code analysis to static software testing," 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), 2014, pp. 1280-1283, doi: 10.1109/WARTIA.2014.6976516

[9] Runeson, Per & Andersson, C. & Thelin, T. & Andrews, A. & Berling, T.. (2006). What Do We Know about Defect Detection Methods?. Software, IEEE. 23. 82 - 90. 10.1109/MS.2006.89.