

CHAPTER 3

TYPES OF TESTING

○ **Static Testing –**

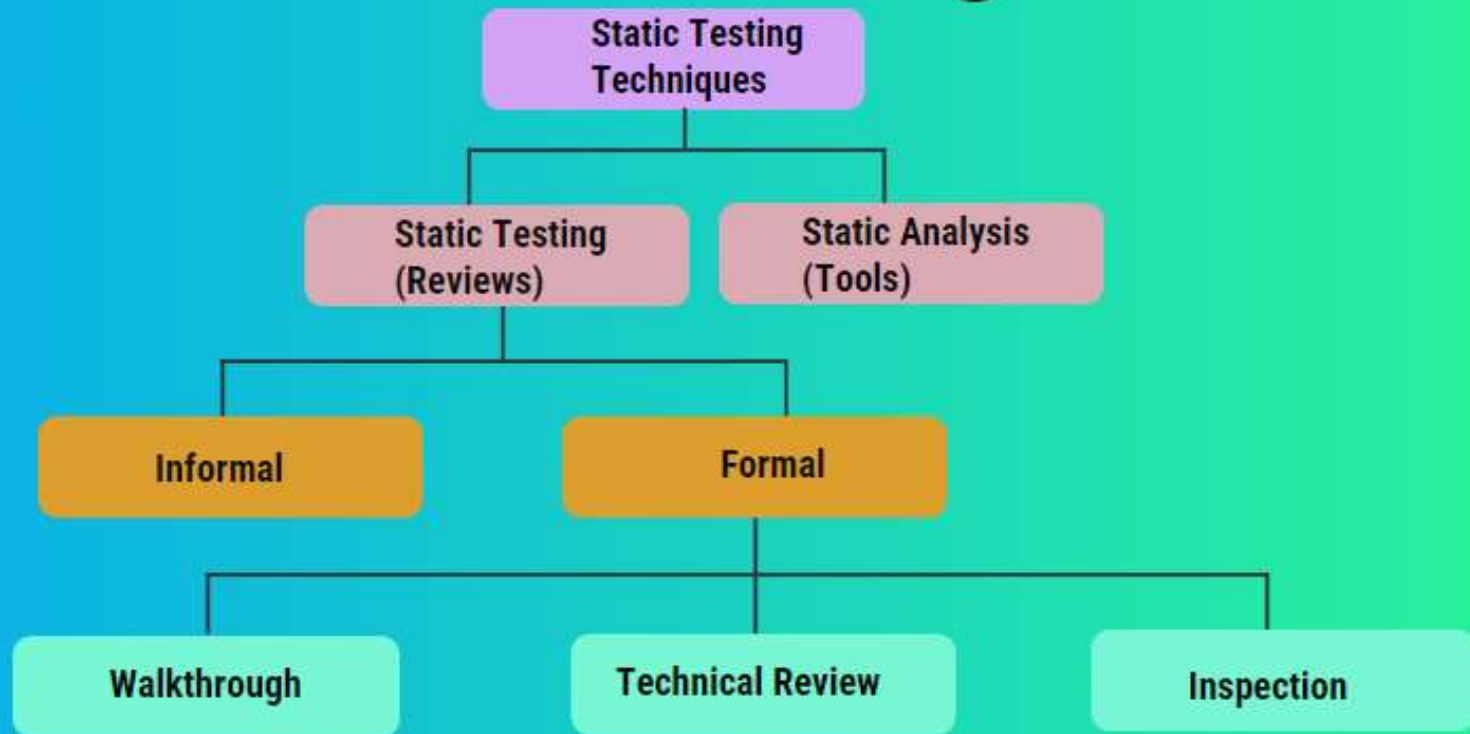
- Testing of a software development artifact
e.g., requirements, design or code,
- without execution of these artifacts
e.g., reviews or static analysis.

○ **Dynamic Testing –**

- Testing that involves the execution of the software of a component or system.



Static Testing



Static Testing	Dynamic Testing
Testing is done without executing the program	Testing is done by executing the program
This testing does verification process	Dynamic testing does validation process
It is about prevention of defects	It is about finding and fixing the defects
Gives assessment of code and documentation	Gives bugs/bottlenecks in the software system.
Involves checklist and process to be followed	Involves test cases for execution
Performed before compilation	Performed after compilation



REVIEW

○ Review –

- An evaluation of a product or project status to ascertain discrepancies from planned to actual component.
- to recommend improvements. Examples include management review, informal review, technical review, inspection, and walkthrough.

1. Informal Review –

A review not based on a formal (documented) procedure

2. Formal Review –

A review characterized by documented procedures and requirements, e.g. inspection.



REVIEW PROCESS

Phases of Formal Review

- Planning
- Kick-off
- Preparation
- Review Meeting
- Rework
- Follow-up



PLANNING

- Begins with '**request for review**' from author to moderator
- Project planning should spare time for review
- Moderator performs '**entry criteria**' check
 - A short check of a product sample by the moderator does not reveal a large number of major defects.
 - The document to be reviewed is available with line numbers.
 - References needed for the inspection are stable and available
- If the document passes the entry check, the moderator and author decide which part of the document to review.
- Moderator determines, in co-operation with the author, the composition of the review team.



KICK-OFF

- An optional step in a review procedure.
- The goal of this meeting is to get everybody on the same wavelength regarding the document under review.
- Everybody commits to the time that will be spent on checking.
- The reviewers receive a short introduction on the objectives of the review and the documents.



PREPARATION

- The participants work individually on the document under review using the related documents, procedures, rules and checklists provided.
- The individual participants identify defects, questions and comments, according to their understanding of the document and role.
- All issues are recorded, preferably using a logging form.



REVIEW MEETING

- The meeting typically consists of the following elements :
 - Logging phase
 - Discussion phase
 - Decision phase



REWORK

- Based on the defects detected, the author will improve the document under review step by step.
- Not every defect that is found leads to rework.
- It is the author's responsibility to judge if a defect has to be fixed.



FOLLOW-UP

- The moderator is responsible for ensuring that satisfactory actions have been taken on
 - All (logged) defects
 - Process improvement suggestions
 - Change requests.



ROLES AND RESPONSIBILITIES

- Within a review team, four types of participants can be distinguished:
 - Moderator,
 - Author,
 - Scribe and
 - Reviewer.
- In addition management needs to play a role in the review process.



MODERATOR

- The moderator (or review leader) **leads** the review process.
- He or she determines, in cooperation with the author,
 - The type of review
 - Approach and
 - The composition of the review team.
- The moderator performs the entry check and the follow-up on the rework, in order to control the quality of the input and output of the review process.
- The moderator
 - Schedules the meeting,
 - Disseminates documents before the meeting,
 - Coaches other team members,
 - Paces the meeting,
 - Leads possible discussions and
 - Stores the data that is collected.



AUTHOR

- Writer of the document under review
- The author's basic goal should be to learn as much as possible with regard to improving the quality of the document, but also to improve his or her ability to write future documents.
- The author's task is to illuminate unclear areas and to understand the defects found.



SCRIBE (RECORDER)

- During the logging meeting, the scribe has to record each defect mentioned and any suggestions for process improvement.



REVIEWER (INSPECTOR)

- The task of the reviewers is to check any material for defects, mostly prior to the meeting.
- The level of thoroughness required depends on the type of review.
- The level of domain knowledge or technical expertise needed by the reviewers also depends on the type of review.



MANAGER

- The manager is involved in the reviews as he or she decides on the execution of reviews, allocates time in project schedules and determines whether review process objectives have been met.
- The manager will also take care of any review training requested by the participants.
- The manager can also be involved in the review itself depending on his or her background, playing the role of a reviewer if this would be helpful.



TYPES OF REVIEW

- Walkthrough
- Technical Review
- Inspection



5 requirements(n test cases)

Desc testcases

Expected input and expected output



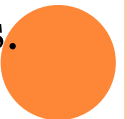
WALKTHROUGH

- A walkthrough is characterized **by the author of the document** under review guiding the participants through the document
- This is especially useful if people from outside the software discipline are present, who are not used to, or cannot easily understand software development documents.
- The content of the document is explained step by step by the author, to reach consensus on changes or to gather information.



WALKTHROUGH

- Within a walkthrough the **author does most of the preparation.**
- The participants, who are selected from different departments and backgrounds, are not required to do a detailed study of the documents in advance
- **A large number of people can participate and this larger audience** can bring a great number of diverse viewpoints regarding the contents of the document being reviewed
- If the audience represents a broad cross-section of skills and disciplines, it can give assurance that no major defects are 'missed' in the walk-through.
- A walkthrough is especially useful for higher-level documents, such as requirement specifications and architectural documents.



TECHNICAL REVIEW

- A technical review is a discussion meeting that focuses on achieving consensus about the **technical content of a document**.
- During technical reviews **defects are found by experts**, who focus on the content of the document.
- The experts that are needed for a technical review are, for example, architects, chief designers and key users.




INSPECTION

- Inspection is the **most formal** review type.
- The document under inspection is prepared and checked thoroughly by the reviewers before the meeting, comparing the work product with its sources and other referenced documents, and using rules and checklists.
- In the inspection meeting the defects found are logged and any discussion is postponed until the discussion phase.
- This makes the inspection meeting a very efficient meeting.



KEY CHARACTERISTICS OF AN INSPECTION

- It is usually led by a trained moderator (certainly not by the author).
 - It uses defined roles during the process.
 - It involves peers to examine the product.
 - Rules and checklists are used during the preparation phase.
 - A separate preparation is carried out during which the product is examined and the defects are found.
 - The defects found are documented in a logging list or issue log.
 - A formal follow-up is carried out by the moderator applying exit criteria.
 - Optionally, a causal analysis step is introduced to address process improvement issues and learn from the defects found.
 - Metrics are gathered and analyzed to optimize the process.
- 

STATIC ANALYSIS

- Analysis of software development artifacts, e.g. requirements or code, carried out without execution of these software development artifacts.
- Static analysis is usually carried out by means of a supporting tool.
- Static analysis is performed on requirements, design or code without actually executing the software artifact being examined.
 - Static analysis is ideally performed before the types of formal reviews.
 - The goal of static analysis is to find defects, whether or not they may cause failures. As with reviews, static analysis finds defects rather than failures.



STATIC ANALYSIS

- Static source code analysis is the process by which software developers check their code for problems and inconsistencies before compiling.
- Organizations can automate the source code analysis process by implementing a tool that
 - Automatically analyzes the entire program
 - Generates charts and
 - Reports that graphically present the analysis results, and
 - Recommends potential resolutions to identified problems.



COMPILER

- For static analysis there are many tools, and most of them focus on software code.
- Static analysis tools are typically used by developers before, and sometimes during, component and integration testing
- Static analysis tools are used by designers during software modelling.
- The tools can show
 - structural attributes (code metrics), such as depth of nesting or cyclomatic number
 - check against coding standards,
 - graphic depictions of control flow, data relationships and the number of distinct paths from one line of code to another.
- **Compiler** - A software tool that translates programs expressed in a high order language into their machine language equivalents.



STATIC ANALYSIS BY TOOLS

Static analysis tools scan the source code and automatically detect errors that typically pass through compilers and become latent problems, including the following:

- Syntax
- Unreachable code
- Unconditional branches into loops
- Undeclared variables
- Uninitialized variables
- Parameter type mismatches
- Uncalled functions and procedures
- Variables used before initialization
- Non-usage of function results
- Possible array bound errors
- Misuse of pointers



STATIC ANALYSIS - EXAMPLES

Search for duplicated pieces of code

- Manually searching for code duplicates in big projects locally can take huge amounts of time, and it is almost impossible to constantly monitor the state of the code base.

#1 ShowProjectDuplicatesAction.java:120

/build-server4idea/src/jetbrains/buildServer/codeInspection

```
116.     final Presentation presentation = anActionEvent.getPresentation();
117.     final Project project = (Project)anActionEvent.getSource();
118.     if (project == null) {
119.         presentation.setVisible(false);
120.         return;
121.     }
```

#2 ShowProjectInspectionsAction.java:128

/build-server4idea/src/jetbrains/buildServer/codeInspection

```
127.     final Presentation presentation = anActionEvent.getPresentation();
128.     final Project project = (Project)anActionEvent.getSource();
129.     if (project == null) {
130.         presentation.setVisible(false);
131.         return;
132.     }
```



CODE STRUCTURE

- There are many different kinds of structural measures
 - Each of which tells us something about the effort required to write the code in the first place
 - To understand the code when making a change
 - To test the code using particular tools or techniques.
- It is often assumed that a large module takes longer to specify, design, code and test than a smaller one.



CODE STRUCTURE

- There are several aspects of code structure to consider:
 - Control flow structure
 - Data flow structure
 - Data structure.



CONTROL FLOW STRUCTURE

- The control flow structure addresses the sequence in which the instructions are executed.
- Control flow analysis can also be used to identify unreachable (dead) code.
- In fact many of the code metrics relate to the control flow structure, e.g. number of nested levels or cyclomatic complexity.



CONTROL FLOW ANALYSIS

- Highlights:
 - Nodes not accessible from start node
 - Infinite loops
 - Multiple entry 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 and other metrics



DATA FLOW STRUCTURE

- Data flow structure follows the trail of a data item as it is accessed and modified by the code.
- Many times, the transactions applied to data are more complex than the instructions that implement them.
- Thus, using data flow measures it is shown how the data act as they are transformed by the program.
- Defects can be found such as referencing a variable with an undefined value and variables that are never used.



DATA STRUCTURE

- Data structure refers to the organization of the data itself, independent of the program.
- When data is arranged as a list, queue, stack, or other well-defined structure, the algorithms for creating, modifying or deleting them are more likely to be well-defined, too.
- Thus, the data structure provides a lot of information about the difficulty in writing programs to handle the data and in designing test cases to show program correctness.
- That is, sometimes a program is complex because it has a complex data structure, rather than because of complex control or data flow.



IMPORTANCE OF STATIC ANALYSIS

- Early detection of defects prior to test execution
- Early warning about suspicious aspects of the code, design or requirements
- Identification of defects not easily found in dynamic testing
- Improved maintainability of code and design since engineers work according to documented standards and rules
- Prevention of defects, provided that engineers are willing to learn from their errors and continuous improvement is practiced.

