
Testing Concepts
Lesson 3: Static Testing
Capgemini

Lesson Objectives

To understand the following topics:

- Types of Testing Techniques
- Differences between Static & Dynamic Testing
- Static Testing Basics
  - Work Products Examined by Static Testing
  - Benefits of Static Testing
- Review Process
  - Work Product Review Process
  - Roles and responsibilities in a formal review
  - Review Types
  - Applying Review Techniques – checklist based Testing in detail.
  - Success Factors for Reviews

Types of Testing Techniques

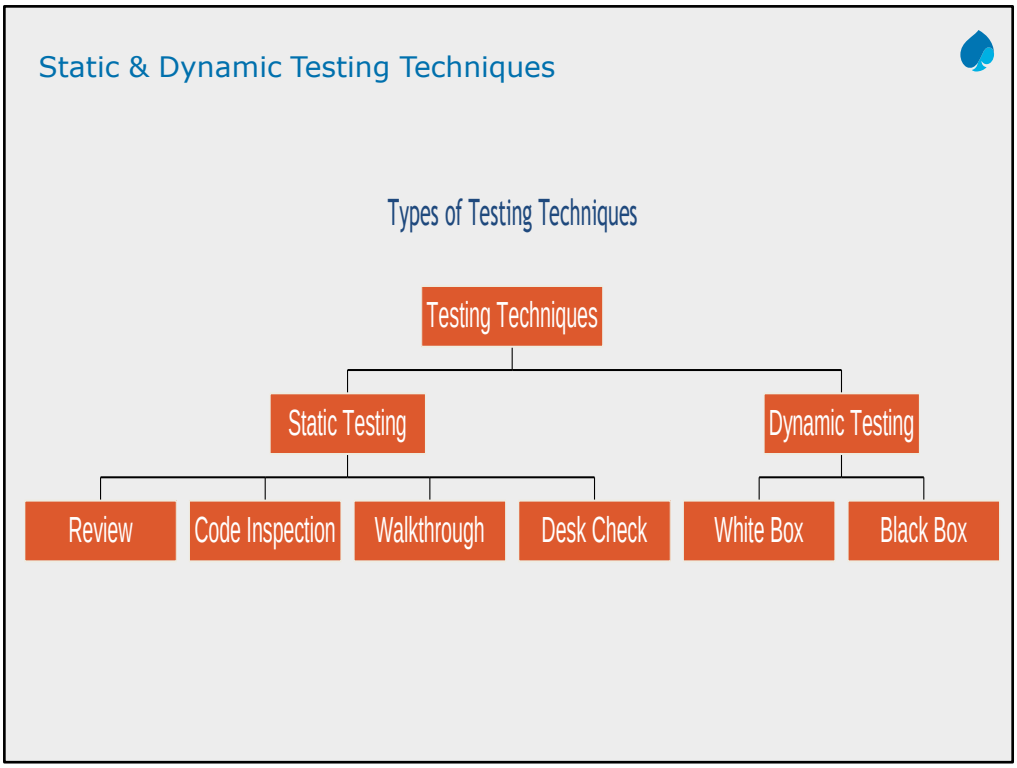
There are two types of Testing Techniques

1. Static Testing

- It is a ***verification*** process
- Testing a software without execution on a computer. Involves just examination/review and evaluation
- It is done to test that software confirms to its SRS i.e. user specified requirements
- It is done for ***preventing*** the defects

2. Dynamic Testing

- It is a ***validation*** process
- Testing software through executing it
- It is done to test that software does what the user really requires
- It is done for ***detecting*** the defects


Static & Dynamic Testing Techniques
Types of Testing Techniques
Testing Techniques
Static Testing
Dynamic Testing
Review
Code Inspection
Walkthrough
Desk Check
White Box
Black Box

Differences between Static and Dynamic Testing

Static Testing	Dynamic Testing
It is the process of confirming whether the software meets its requirement specification.	It is the process of confirming whether the software meets user requirements.
Examples : Inspections, walkthroughs and reviews.	Examples : structural testing, black-box testing, integration testing, acceptance testing.
It is the process of inspecting without executing on computer.	It is the process of testing by executing on computer.
It is conducted to prevent defects.	It is conducted to correct the defects
It can be done before compilation	It takes place only after compilation and linking.

One main distinction is that static testing finds defects in work products directly rather than identifying failures caused by defects when the software is run. A defect can reside in a work product for a very long time without causing a failure. The path where the defect lies may be rarely exercised or hard to reach, so it will not be easy to construct and execute a dynamic test that encounters it. Static testing may be able to find the defect with much less effort.
Another distinction is that static testing can be used to improve the consistency and internal quality of work products, while dynamic testing typically focuses on externally visible behaviors.

3.1 Static Testing Basics

- static testing relies on the manual examination of work products (i.e., reviews) or tool-driven evaluation of the code or other work products (i.e., static analysis).
- Both types of static testing assess the code or other work product being tested without actually executing the code or work product being tested.
- Static analysis is important for safety-critical computer systems (e.g., aviation, medical, or nuclear software).
- Static analysis is an important part of security testing.
- Static analysis is also often incorporated into automated build and delivery systems, for example in Agile development, continuous delivery, and continuous deployment.

Static Testing Basics

- These static techniques rely on manual examinations (Reviews) and automated Analysis (Static Analysis) without execution.
- Static Testing helps weed out many errors/bugs at an early stage
- Static Testing lays strict emphasis on conforming to specifications
- Static Testing can discover dead codes, infinite loops, uninitialized and unused variables, standard violations and is effective in finding 30-70% of errors.

3.1.1 Work Products examined by Static Testing

- Specifications, including business requirements, functional requirements, and security requirements
- Epics, user stories, and acceptance criteria
- Architecture and design specifications
- Code
- Testware, including test plans, test cases, test procedures, and automated test scripts
- User guides
- Web pages
- Contracts, project plans, schedules, and budgets
- Models, such as activity diagrams, which may be used for Model-Based testing

Static analysis can be applied efficiently to any work product with a formal structure (typically code or models) for which an appropriate static analysis tool exists. Static analysis can even be applied with tools that evaluate work products written in natural language such as requirements (e.g., checking for spelling, grammar, and readability).

3.1.2 Benefits of Static Testing

- Detecting and correcting defects more efficiently and prior to dynamic testing.
- Identifying defects which are not easily found by dynamic testing
- Preventing defects in design or coding by uncovering inconsistencies, ambiguities, contradictions, omissions, inaccuracies, and redundancies in requirements
- Increasing development productivity (improved design, more maintainable code)
- Reducing development cost and time
- Reducing testing cost and time
- Reducing total cost of quality over the software's lifetime, due to fewer failures later in the lifecycle or after delivery into operation
- Improves communication between team members.
- Increased awareness of quality issues and early feedback on quality.

Benefits of Static Testing :

- When applied early in the SDLC, static testing enables the early detection of defects before dynamic testing is performed (e.g., in requirements or design specifications reviews, product backlog refinement, etc.).
- Defects found early are often much cheaper to remove than defects found later in the lifecycle, especially compared to defects found after the software is deployed and in active use.
- Using static testing techniques to find defects and then fixing those defects promptly is almost always much cheaper for the organization than using dynamic testing to find defects in the test object and then fixing them, especially when considering the additional costs associated with updating other work products and performing confirmation and regression testing.

Typical Defects found during Static Testing

- Requirement defects (e.g., inconsistencies, ambiguities, contradictions, omissions, inaccuracies, and redundancies)
- Design defects (e.g., inefficient algorithms or database structures, high coupling, low cohesion)
- Coding defects (e.g., variables with undefined values, variables that are declared but never used, unreachable code, duplicate code)
- Deviations from standards (e.g., lack of adherence to coding standards)
- Incorrect interface specifications (e.g., different units of measurement used by the calling system than by the called system)
- Security vulnerabilities (e.g., susceptibility to buffer overflows)
- Gaps or inaccuracies in test basis traceability or coverage (e.g., missing tests for an acceptance criterion)

Compared with dynamic testing, the above typical defects are easier and cheaper to find and fix through static testing.
Moreover, most types of maintainability defects can only be found by static testing (e.g., improper modularization, poor reusability of components, code that is difficult to analyze and modify without introducing new defects).

3.2 Review Process

Reviews vary from informal to formal.

- Informal reviews are characterized by not following a defined process and not having formal documented output.
- Formal reviews are characterized by team participation, documented results of the review, and documented procedures for conducting the review. The formality of a review process is related to factors such as the SDLC model, the maturity of the development process, the complexity of the work product to be reviewed, any legal or regulatory requirements, and/or the need for an audit trail.
- The focus of a review depends on the agreed objectives of the review (e.g., finding defects, gaining understanding, educating participants such as testers and new team members, or discussing and deciding by consensus).

3.2.1 Work Product Review Process

- The review process comprises the following main activities:
- Planning
- Initiate review
- Individual review (i.e., individual preparation)
- Issue communication and analysis
- Fixing and Reporting

The review process comprises the following main activities:

Planning

- Defining the scope, which includes the purpose of the review, what documents or parts of documents to review, and the quality characteristics to be evaluated
- Estimating effort and timeframe
- Identifying review characteristics such as the review type with roles, activities, and checklists
- Selecting the people to participate in the review and allocating roles
- Defining the entry and exit criteria for more formal review types (e.g., inspections)
- Checking that entry criteria are met (for more formal review types)

Initiate review

- Distributing the work product (physically or by electronic means) and other material, such as issue log forms, checklists, and related work products
- Explaining the scope, objectives, process, roles, and work products to the participants
- Answering any questions that participants may have about the review

Individual review (i.e., individual preparation)

- Reviewing all or part of the work product
- Noting potential defects, recommendations, and questions Issue communication and analysis
- Communicating identified potential defects (e.g., in a review meeting)
- Analyzing potential defects, assigning ownership and status to them
- Evaluating and documenting quality characteristics
- Evaluating the review findings against the exit criteria to make a review decision (reject; major changes needed; accept, possibly with minor changes)

Fixing and reporting

- Creating defect reports for those findings that require changes
- Fixing defects found (typically done by the author) in the work product reviewed
- Communicating defects to the appropriate person or team (when found in a work product related to the work product reviewed)
- Recording updated status of defects (in formal reviews), potentially including the agreement of the comment originator
- Gathering metrics (for more formal review types)
- Checking that exit criteria are met (for more formal review types)
- Accepting the work product when the exit criteria are reached

Code Review Checklist used in Review Process

Data Reference Errors
- Is a variable referenced whose value is unset or uninitialized?

Data Declaration Errors
- Have all variables been explicitly declared?
- Are variables properly initialized in declaration sections?

Computation errors
- Are there any computations using variables having inconsistent data types?
- Is there any mixed mode computations?

Comparison errors
- Are there any comparisons between variables having inconsistent data types?

Control Flow errors
- Will every loop eventually terminate?
- Is it possible that, because of condition upon entry, a loop will never execute?

Interface errors
- Does the number of parameters received by these module equals the number of arguments sent by calling modules?
- Also is the order correct?

Input/output errors
- All I/O conditions handled correctly?

ERROR CHECKLIST FOR INSPECTION

1. Data Reference Errors

For each array reference, is each subscript value within defined bounds?
Dangling reference problem: arises when the lifetime of a pointer is greater than the lifetime of a referenced storage.
If a data structure is referenced in multiple procedures or subroutines, is the structure defined identically in each procedure?

2. Data Declaration errors

Is each variable has been assigned correct length, type, and storage class?
Are there any variables with similar names (not an error but may have been confused with in the program)?

3. Computation errors

Are there any computations using same data types but different lengths?
Is an underflow or overflow occurs during computation?
Division by zero and square root of a negative number errors.
Are the order of evaluation and precedence of operators correct?

4. Comparison errors:

Are there any mixed mode computations or comparisons between variables of different lengths?
Is the comparison operator correct? Does each Boolean expression state what it is supposed to do? Are the operands of a Boolean operator Boolean?

3.2.2 Roles & Responsibilities in a Formal Review

- **Author**
  - Creates the work product under review
  - Fixes defects in the work product under review (if necessary)
- **Management**
  - Is responsible for review planning
  - Decides on the execution of reviews
  - Assigns staff, budget, and time
  - Monitors ongoing cost-effectiveness
  - Executes control decisions in the event of inadequate outcomes
- **Facilitator (often called moderator)**
  - Ensures effective running of review meetings (when held)
  - Mediates, if necessary, between the various points of view
  - Is often the person upon whom the success of the review depends

3.2.2 Roles & Responsibilities in a Formal Review (Cont..)

- **Review leader**
  - Takes overall responsibility for the review
  - Decides who will be involved and organizes when and where it will take place
- **Reviewers**
  - May be SME, persons working on the project, stakeholders with an interest in the work product, and/or individuals with specific technical or business backgrounds
  - Identify potential defects in the work product under review
  - May represent different perspectives (e.g., tester, programmer, user, operator, business analyst, usability expert, etc.)
- **Scribe (or recorder)**
  - Collates potential defects found during the individual review activity
  - Records new potential defects, open points, and decisions from the review meeting (when held)

3.2.3 Review Types

- **Self Review**
- **Informal Review**
- **Walkthrough**
- **Technical Review**
- **Inspection**

Self Review

- Self review is done by the person who is responsible for a particular program code
- It is more of reviewing the code in informal way
- It is more like who writes the code, understands it better
- Self review is to be done by the programmer when he builds a new code
- There are review checklists that helps programmer to verify with the common errors regarding the program code

Informal Review

- Examples: buddy check, pairing, pair review
- Main purpose: detecting potential defects
- Possible additional purposes: generating new ideas or solutions, quickly solving minor problems
- Not based on a formal (documented) process
- May not involve a review meeting
- May be performed by a colleague of the author (buddy check) or by more people
- Results may be documented
- Varies in usefulness depending on the reviewers
- Use of checklists is optional
- Very commonly used in Agile development

Code Walkthrough

- Main purposes: find defects, improve the software product, consider alternative implementations, evaluate conformance to standards and specifications
- Possible additional purposes: exchanging ideas about techniques or style variations, training of participants, achieving consensus
- Individual preparation before the review meeting is optional
- Review meeting is typically led by the author of the work product
- Scribe is mandatory
- Use of checklists is optional
- May take the form of scenarios, dry runs, or simulations
- Potential defect logs and review reports may be produced
- May vary in practice from quite informal to very formal

Code Walkthrough is a set of procedures and error detection techniques for group reading.
It is a group activity.
In Walkthrough meeting, 3-5 people are involved. Out of the three, one is moderator, the second is Secretary who is responsible for recording all the errors and the third person plays a role of Test Engineer.
Solutions are also suggested by team members.
Walkthrough helps in :

Approach to Solution
Find omission of requirements
Style / Concepts Issues
Detect Defects
Educate Team Members

Technical Review

- Main purposes: gaining consensus, detecting potential defects
- Other purposes: evaluating quality and building confidence in the work product, generating new ideas, motivating and enabling authors to improve future work products, considering alternative implementations
- Reviewers should be technical peers of the author, and technical experts in the same or other disciplines
- Individual preparation before the review meeting is required
- Review meeting is optional, ideally led by a trained facilitator (not the author)
- Scribe is mandatory, ideally not the author
- Use of checklists is optional
- Potential defect logs and review reports are typically produced

Code Inspection

- Main purposes: detecting potential defects, evaluating quality and building confidence in the work product, preventing future similar defects through author learning and root cause analysis
- Possible further purposes: motivating and enabling authors to improve future work products and the software development process, achieving consensus
- Follows a defined process with formal documented outputs, based on rules and checklists
- An inspection team usually consists of clearly defined roles, such as those specified in section 3.2.2.

Code Inspection :
Code inspection is a set of procedures and error detection techniques for group code reading.

Involves reading or visual inspection of a program by a team of people, hence it is a

group activity.

The objective is to find errors but not solutions to the errors

Code Inspection (Cont..)

- Individual preparation before the review meeting is required
- Reviewers are either peers of the author or experts in other disciplines that are relevant to the work product.
- Specified entry and exit criteria are used
- Scribe is mandatory
- Review meeting is led by a trained facilitator (not the author)
- Author cannot act as the review leader, reader, or scribe
- Potential defect logs and review report are produced
- Metrics are collected and used to improve the entire software development process, including the inspection process

Code inspection focuses on discovering errors and not correcting them.

Code Inspection Process :

Before the Inspection

The moderator distributes the program's listing and design specification to the group well in advance of the inspection session

During the inspection

The programmer narrates the logic of the program, statement by statement

During the discourse, questions are raised and pursued to determine if errors exist

The program is analyzed with respect to a check list of historically common programming errors

Desk Checking (Peer Review)

- Human error detection technique
- Viewed as a one person inspection or walkthrough
- A person reads a program and checks it with respect to an error list and/or walks test data through it
- Less effective technique
- Best performed by the person other than the author of the program

It is the dry run of the program. It is completely informal process. Desk checking is best performed by a person other than the author of the program.
E.g. Two programmers might swap the program rather than desk checking their own programs.
It is less effective than inspection and walkthrough of the code.

3.2.4 Applying Review Techniques

There are a number of review techniques that can be applied during the individual review (i.e., individual preparation) activity to uncover defects.

- **Ad hoc**
- **Checklist-based**
- **Scenarios and dry runs**
- **Role-based**
- **Perspective-based**

Ad hoc :
In an ad hoc review, reviewers are provided with little or no guidance on how this task should be performed. Reviewers often read the work product sequentially, identifying and documenting issues as they encounter them. Ad hoc reviewing is a commonly used technique needing little preparation. This technique is highly dependent on reviewer skills and may lead to many duplicate issues being reported by different reviewers.

Checklist-based :
A checklist-based review is a systematic technique, whereby the reviewers detect issues based on checklists that are distributed at review initiation (e.g., by the facilitator). A review checklist consists of a set of questions based on potential defects, which may be derived from experience. Checklists should be specific to the type of work product under review and should be maintained regularly to cover issue types missed in previous reviews.
The main advantage of the checklist-based technique is a systematic coverage of typical defect types. Care should be taken not to simply follow the checklist in individual reviewing, but also to look for defects outside the checklist.

Scenarios and dry runs
In a scenario-based review, reviewers are provided with structured guidelines on how to read through the work product. A scenario-based approach supports reviewers in performing “dry runs” on the work product based on expected usage of the work product (if the work product is documented in a suitable format such as use cases). These scenarios provide reviewers with better guidelines on how to identify specific defect types than simple checklist entries. As with checklist-based reviews, in order not to miss other defect types (e.g., missing features), reviewers should not be constrained to the documented scenarios.

3.2.5 Success Factors for Reviews

Organizational Success Factors for Reviews

- Each review has clear objectives, defined during review planning, and used as measurable exit criteria
- Review types are applied which are suitable to achieve the objectives and are appropriate to the type and level of software work products and participants
- Any review techniques used, such as checklist-based or role-based reviewing, are suitable for effective defect identification in the work product to be reviewed
- Any checklists used address the main risks and are up to date
- Large documents are written and reviewed in small chunks, so that quality control is exercised by providing authors early and frequent feedback on defects
- Participants have adequate time to prepare
- Reviews are scheduled with adequate notice
- Management supports the review process (e.g., by incorporating adequate time for review activities in project schedules)

3.2.5 Success Factors for Reviews (Cont..)

People-related success factors for reviews include:

- The right people are involved to meet the review objectives, for example, people with different skill sets or perspectives, who may use the document as a work input
- Testers are seen as valued reviewers who contribute to the review and learn about the work product, which enables them to prepare more effective tests, and to prepare those tests earlier
- Participants dedicate adequate time and attention to detail
- Reviews are conducted on small chunks, so that reviewers do not lose concentration during individual review and/or the review meeting (when held)

3.2.5 Success Factors for Reviews (Cont..)

People-related success factors for reviews include:

- Defects found are acknowledged, appreciated, and handled objectively
- The meeting is well-managed, so that participants consider it a valuable use of their time
- The review is conducted in an atmosphere of trust; the outcome will not be used for the evaluation of the participants
- Participants avoid body language and behaviors that might indicate boredom, exasperation, or hostility to other participants
- Adequate training is provided, especially for more formal review types such as inspections
- A culture of learning and process improvement is promoted

Summary

In this lesson, you have learnt:

- Basics of Static Testing
- Review Process
- Various Review Types
- Applying Review Techniques
- Success Factors for Reviews

Review Question

Question 1: _____ testing can discover dead codes

Question 2: The objective of walkthrough is to find errors but not solutions. (T/F)

Question 3: State the various Review Types

Question 4: State the Success factors for Review process