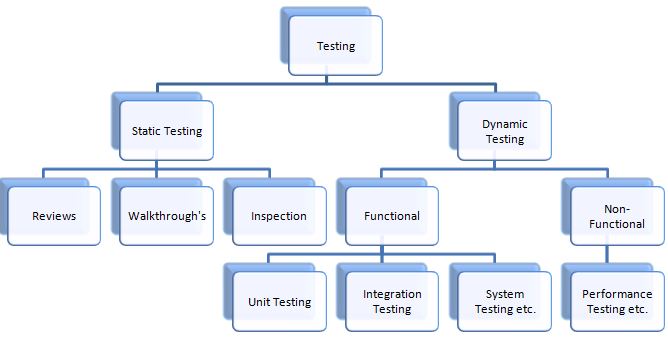
# **Static vs Dynamic Testing**

## **📖 Introduction**

Software testing is broadly classified into Static Testing and Dynamic Testing:

| **Static Testing** | **Dynamic Testing** |
| --- | --- |
| Examines code/docs without execution. | Tests software by executing it. |
| Finds early-stage defects. | Catches runtime errors. |
| Examples: Reviews, walkthroughs. | Examples: Unit tests, system tests. |

✅ Goal: Improve quality by combining both approaches.



## **📜 Static Testing**

### **Definition**

Analyzing software without running it to detect defects in requirements, design, and code.

### **Types of Static Testing**

1. Reviews
   * Informal Reviews: Quick team discussions.
   * Formal Inspections: Structured, documented (e.g., Fagan Inspection).
2. Walkthroughs
   * Author explains the code/doc to peers.
3. Static Code Analysis
   * Tools scan code for vulnerabilities, bugs, and style issues.
   * Tools: SonarQube, ESLint, Checkmarx.
4. Requirement Analysis
   * Checks for ambiguity, completeness, and testability.

### **Pros & Cons**

| **Pros** | **Cons** |
| --- | --- |
| ✅ Early defect detection (cheaper to fix). | ❌ Cannot find runtime issues. |
| ✅ Improves documentation quality. | ❌ Manual reviews are time-consuming. |

### **Best Use Cases**

✔ Before coding starts (Requirement docs).  
✔ Early development phase (Design/code reviews).

## **⚡ Dynamic Testing**

### **Definition**

Testing software by executing it with test inputs and validating outputs.

### **Types of Dynamic Testing**

1. Functional Testing
   * Validates features against requirements.
   * Examples: Unit, Integration, System, UAT.
2. Non-Functional Testing
   * Tests performance, security, usability.
   * Examples: Load testing, Penetration testing.
3. Regression Testing
   * Ensures new changes don’t break old features.

### **Techniques**

* Black-Box Testing (No code access).
* White-Box Testing (Code-level validation).

### **Pros & Cons**

| **Pros** | **Cons** |
| --- | --- |
| ✅ Finds runtime bugs (crashes, memory leaks). | ❌ Requires executable code. |
| ✅ Validates real-world behavior. | ❌ More expensive than static testing. |

### **Best Use Cases**

✔ After code is developed (Unit testing).  
✔ Pre-release (System/UAT testing).

## **🆚 Key Differences**

| **Aspect** | **Static Testing** | **Dynamic Testing** |
| --- | --- | --- |
| Execution | No code execution. | Code is executed. |
| Stage | Early (Requirements/Design). | Later (Development/Testing). |
| Defects Found | Syntax errors, design flaws. | Runtime errors, performance issues. |
| Automation | Limited (Mostly manual). | Highly automatable. |
| Cost | Low (Early detection). | High (Requires test envs). |

✅ Example:

* Static: Reviewing a login feature’s design doc.
* Dynamic: Testing the login feature with valid/invalid inputs.

## **⏰ When to Use Each?**

### **Static Testing is Better For:**

* Catching typos in requirements.
* Improving code readability (peer reviews).
* Security vulnerabilities (static code analysis).

### **Dynamic Testing is Better For:**

* Validating user flows (e.g., checkout process).
* Performance bottlenecks (load testing).
* Real-world scenarios (browser/device testing).

### **Combined Approach = Maximum Coverage**

* Shift-Left Testing: Apply static testing early.
* Continuous Testing: Run dynamic tests in CI/CD.

## **🏆 Best Practices**

### **For Static Testing**

✔ Use checklists for reviews (ensure consistency).  
✔ Automate code analysis (SonarQube, ESLint).  
✔ Involve cross-functional teams (devs, QA, BAs).

### **For Dynamic Testing**

✔ Prioritize test cases (risk-based testing).  
✔ Combine manual + automated testing.  
✔ Monitor in production (A/B testing, logs).

### **For Both**

✔ Maintain traceability (link tests to requirements).  
✔ Document defects with severity/priority.

## **🌍 Real-World Examples**

### **Case 1: NASA Mars Climate Orbiter (Static Testing Failure)**

* Defect: Metric/Imperial unit mismatch in code.
* Root Cause: No rigorous design review.
* Loss: $327 million spacecraft.

### **Case 2: Knight Capital Crash (Dynamic Testing Gap)**

* Defect: Untested code deployed to production.
* Root Cause: No regression testing.
* Loss: $460 million in 45 minutes.

### **Case 3: GitHub’s Secure Code (Static + Dynamic Success)**

* Static: Code reviews + SAST (Semgrep).
* Dynamic: DAST (OWASP ZAP) + Pen testing.
* Result: Reduced vulnerabilities by 60%.

## **📚 References**

* [ISTQB Static vs Dynamic Testing](https://www.istqb.org/)
* [OWASP Testing Guide](https://owasp.org/www-project-web-security-testing-guide/)
* [Microsoft SDLC Testing](https://docs.microsoft.com/en-us/devops/develop/test/)

🔹 Conclusion:

* Static Testing = Prevention (Early defect detection).
* Dynamic Testing = Validation (Real-world verification).
* Use both for cost-effective, high-quality software.