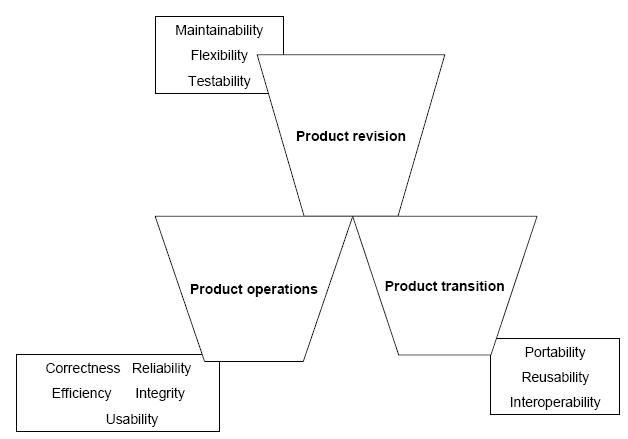
A 21 : <https://msritse2012.wordpress.com/2013/01/27/quality-models-in-software-engineering/>

**McCall’s Quality model (1977**)  
Also called as General Electrics Model. This model was mainly developed for US military to bridge the gap between users and developers. It mainly has 3 major representations for defining and identifying the quality of a software product, namely  
Product Revision : This identifies quality factors that influence the ability to change the software product.  
(1) Maintainability : Effort required to locate and fix a fault in the program within its operating environment.  
(2) Flexibility : The ease of making changes required as dictated by business by changes in the operating environment.  
(3) Testability : The ease of testing program to ensure that it is error-free and meets its specification, i.e, validating the software requirements.  
Product Transition : This identifies quality factors that influence the ability to adapt the software to new environments.  
(1) Portability : The effort required to transfer a program from one environment to another.  
(2) Re-usability : The ease of reusing software in a different context.  
(3) Interoperability: The effort required to couple the system to another system.  
Product Operations : This identifies quality factors that influence the extent to which the software fulfills its specification.  
(1) Correctness : The extent to which a functionality matches its specification.  
(2) Reliability : The systems ability not to fail/ the extent to which the system fails.  
(3) Efficiency : Further categorized into execution efficiency and storage efficiency and generally means the usage of system resources, example: processor time, memory.  
(4) Integrity : The protection of program from unauthorized access.  
(5) Usability : The ease of using software.



A23 : <http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/>

V- model means Verification and Validation model. Just like the [waterfall model](http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/), the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins.  Testing of the product is planned in parallel with a corresponding phase of development.

The various phases of the V-model are as follows:

**Requirements** like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a [system test](http://istqbexamcertification.com/what-is-system-testing/) plan is created.  The test plan focuses on meeting the functionality specified in the requirements gathering.

**The high-level design (HLD)** phase focuses on system architecture and design. It provide overview of solution, platform, system, product and service/process. An [integration test](http://istqbexamcertification.com/what-is-integration-testing/) plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

**The low-level design** **(LLD)** phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. Component tests are created in this phase as well.

**The implementation** phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

**Coding:** This is at the bottom of the V-Shape model. Module design is converted into code by developers.

**Advantages of V-model:**

* Simple and easy to use.
* Testing activities like planning, [test designing](http://istqbexamcertification.com/what-is-test-design-or-how-to-specify-test-cases/) happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
* Proactive defect tracking – that is defects are found at early stage.
* Avoids the downward flow of the defects.
* Works well for small projects where requirements are easily understood.

**Disadvantages of V-model:**

* Very rigid and least flexible.
* Software is developed during the implementation phase, so no early prototypes of the software are produced.
* If any changes happen in midway, then the test documents along with requirement documents has to be updated.

**When to use the V-model:**

* The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
* The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.



A 25 : <http://www.tutorialspoint.com/software_testing_dictionary/white_box_testing.htm>

**What is White Box Testing?**

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

**White Box Testing Techniques:**

* **Statement Coverage -** This technique is aimed at exercising all programming statements with minimal tests.
* **Branch Coverage -** This technique is running a series of tests to ensure that all branches are tested at least once.
* **Path Coverage -** This technique corresponds to testing all possible paths which means that each statement and branch is covered.

**Advantages of White Box Testing:**

* Forces test developer to reason carefully about implementation.
* Reveals errors in "hidden" code.
* Spots the Dead Code or other issues with respect to best programming practices.

**Disadvantages of White Box Testing:**

* Expensive as one has to spend both time and money to perform white box testing.
* Every possibility that few lines of code are missed accidentally.
* In-depth knowledge about the programming language is necessary to perform white box testing.

<http://www.tutorialspoint.com/software_testing_dictionary/basis_path_testing.htm>

## What is Basis Path Testing?

Basis path testing, a structured testing or white box testing technique used for designing test cases intended to examine all possible paths of execution at least once. Creating and executing tests for all possible paths results in 100% statement coverage and 100% branch coverage.

## Example:

Function fn\_delete\_element (int value, int array\_size, int array[])

{

1 int i;

location = array\_size + 1;

2 for i = 1 to array\_size

3 if ( array[i] == value )

4 location = i;

end if;

end for;

5 for i = location to array\_size

6 array[i] = array[i+1];

end for;

7 array\_size --;

}

## Steps to Calculate the independent paths

**Step 1 :** Draw the Flow Graph of the Function/Program under consideration as shown below:

**Step 2 :** Determine the independent paths.

Path 1: 1 - 2 - 5 - 7

Path 2: 1 - 2 - 5 - 6 - 7

Path 3: 1 - 2 - 3 - 2 - 5 - 6 - 7

Path 4: 1 - 2 - 3 - 4 - 2 - 5 - 6 - 7

A 27 : <http://en.wikipedia.org/wiki/Equivalence_partitioning>

**Equivalence partitioning** (also called **Equivalence Class Partitioning** or **ECP**[[1]](http://en.wikipedia.org/wiki/Equivalence_partitioning" \l "cite_note-1)) is a [software testing](http://en.wikipedia.org/wiki/Software_testing) technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived. In principle, test cases are designed to cover each partition at least once. This technique tries to define test cases that uncover classes of errors, thereby reducing the total number of test cases that must be developed. An advantage of this approach is reduction in the time required for testing a software due to lesser number of test cases.

<http://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/>

← [What is your job profile in software testing? [POLL]](http://www.softwaretestinghelp.com/what-is-your-job-profile-in-software-testing/)

[Practical approaches to improving your testing by maximizing code coverage [White paper for download]](http://www.softwaretestinghelp.com/practical-approaches-to-improving-your-testing-by-maximizing-code-coverage/) →

# What is Boundary value analysis and Equivalence partitioning?

Posted In | [Questions & answers](http://www.softwaretestinghelp.com/category/questions-answers/), [Testing Concepts](http://www.softwaretestinghelp.com/category/testing-concepts/), [Testing Interview questions](http://www.softwaretestinghelp.com/category/software-testing-interview-questions-with-answers/)

**Boundary value analysis and Equivalence partitioning, explained with simple example:**

Boundary value analysis and equivalence partitioning both are test case design strategies in black box testing.

**Equivalence Partitioning:**

In this method the input domain data is divided into different equivalence data classes. This method is typically used **to reduce the total number of test case**s to a finite set of testable test cases, still covering maximum requirements.

In short it is the process of taking all possible test cases and placing them into classes. One test value is picked from each class while testing.

**E.g.:** If you are testing for an input box accepting numbers from 1 to 1000 then there is no use in writing thousand test cases for all 1000 valid input numbers plus other test cases for invalid data.

Using equivalence partitioning method above test cases can be divided into three sets of input data called as classes. Each test case is a representative of respective class.

So in above example we can divide our test cases into three equivalence classes of some valid and invalid inputs.

**Test cases for input box accepting numbers between 1 and 1000 using Equivalence Partitioning:**  
**1)** One input data class with all valid inputs. Pick a single value from range 1 to 1000 as a valid test case. If you select other values between 1 and 1000 then result is going to be same. So one test case for valid input data should be sufficient.

**2)** Input data class with all values below lower limit. I.e. any value below 1, as a invalid input data test case.

**3)** Input data with any value greater than 1000 to represent third invalid input class.

So using equivalence partitioning you have categorized all possible test cases into three classes. Test cases with other values from any class should give you the same result.

We have selected one representative from every input class to design our test cases. Test case values are selected in such a way that largest number of attributes of equivalence class can be exercised.

Equivalence partitioning uses fewest test cases to cover maximum requirements.

**Boundary value analysis:**

It’s widely recognized that input values at the extreme ends of input domain cause more errors in system. More application **errors occur at the boundaries** of input domain. ‘Boundary value analysis’ testing technique is used to identify errors at boundaries rather than finding those exist in center of input domain.

Boundary value analysis is a next part of Equivalence partitioning for designing test cases where test cases are selected at the edges of the equivalence classes.

**Test cases for input box accepting numbers between 1 and 1000 using Boundary value analysis:**  
**1)** Test cases with test data exactly as the input boundaries of input domain i.e. values 1 and 1000 in our case.

A 29 : <http://en.wikipedia.org/wiki/Software_quality>

In the context of [software engineering](http://en.wikipedia.org/wiki/Software_engineering), **software quality** refers to two related but distinct notions that exist wherever [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) is defined in a business context:

* Software functional quality reflects how well it complies with or conforms to a given design, based on [functional requirements](http://en.wikipedia.org/wiki/Functional_requirements) or specifications. That attribute can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile [product](http://en.wikipedia.org/wiki/Product_%28business%29);[[1]](http://en.wikipedia.org/wiki/Software_quality#cite_note-1)
* Software structural quality refers to how it meets [non-functional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly.

A 31 : A test design technique basically helps us to select a good set of tests from the total number of all possible tests for a given system. There are many different types of software testing technique, each with its own strengths and weaknesses. Each individual technique is good at finding particular types of defect and relatively poor at finding other types.

For example, a technique that explores the upper and lower limits of a single input range is more likely to find boundary value defects than defects associated with combinations of inputs. Similarly, testing performed at different stages in the software development life cycle will find different types of defects; component testing is more likely to find coding logic defects than system design defects.

Referring to the figure 4.1 each testing technique falls into one of a number of different categories. Broadly speaking there are two main categories:

1. **Static technique**
2. **Dynamic technique**

* *Specification-based (black-box, also known as behavioral techniques)*
* *Structure-based (white-box or structural techniques)*
* *Experience- based*

 Dynamic techniques are subdivided into three more categories: specification-based (black-box, also known as behavioral techniques), structure-based (white-box or structural techniques) and experience- based. Specification-based techniques include both functional and nonfunctional techniques (i.e. quality characteristics).

### Static testing techniques

Static testing techniques do not execute the code being examined and are generally used before any tests are executed on the software. They could be called non-execution techniques.    
  
Most static testing techniques can be used to 'test' any form of document including source code, design documents and models, functional specifications and requirement specifications.    
  
However, 'static analysis' is a tool-supported type of static testing that concentrates on testing formal languages and so is most often used to statically test source code.

### Specification-based (black-box) testing techniques

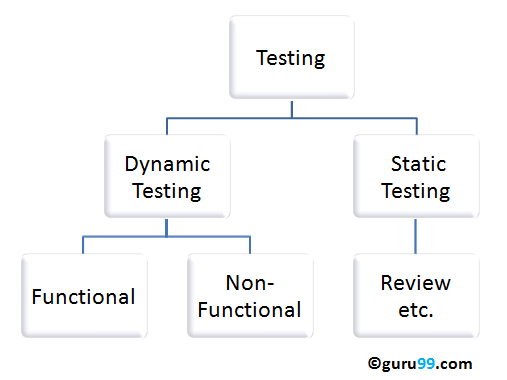
These are also known as 'black-box' or input/output-driven testing techniques because they view the software as a black-box with inputs and outputs, but they have no knowledge of how  
the system or component is structured inside the box.   
  
In essence, the tester is concentrating on what the software does, not how it does it. Functional testing is concerned with what the system does, its features or functions. Non-functional testing is concerned with examining how well the system does something, rather than what it does.   
  
Non-functional aspects (also known as quality characteristics or quality attributes) include performance, usability, portability, maintainability, etc. Techniques to test these non-functional aspects are less procedural and less formalized than those of other cate-gories as the actual tests are more dependent on the type of system, what it does and the resources available for the tests. 

### Structure-based (white-box) testing techniques

Structure-based testing techniques (which are also dynamic rather than static) use the internal structure of the software to derive test cases. They are commonly called 'white-box' or 'glass-box' techniques (implying you can see into the system) since they require knowledge of how the software is implemented, that is, how it works.

* Ex: a structural technique may be concerned with exercising loops in the software. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.

### Experience-based testing techniques

In experience-based techniques, people's knowledge, skills and background are a prime contributor to the test conditions and test cases. The experience of both technical and business people is important, as they bring different perspectives to the test analysis and design process.   
  
Due to previous experience with similar systems, they may have insights into what could go wrong, which is very useful for testing. 

A 33 :

 A defect is an error or a bug, in the application which is created. A programmer while designing and building the software can make mistakes or error. These mistakes or errors mean that there are flaws in the software. These are called defects.

 When actual result deviates from the expected result while testing a software application or product then it results into a defect. Hence, any deviation from the specification mentioned in the product functional specification document is a defect. In different organizations it’s called differently like bug, issue, incidents or problem.

 When the result of the software application or product does not meet with the end user expectations or the software requirements then it results into a Bug or Defect. These defects or bugs occur because of an error in logic or in coding which results into the failure or unpredicted or unanticipated results.

A 35 :

**CMM (Capability Maturity Model)** is a model of process maturity for software development - an evolutionary model of the progress of a company’s abilities to develop software.

The key concept of the standard is organizational maturity. A mature organization has clearly defined **procedures** for software development and project management. These procedures are adjusted and perfected as required.

The CMM model defines five levels of organizational maturity:

1. **Initial level** is a basis for comparison with the next levels. In an organization at the initial level, conditions are not stable for the development of quality software. The results of any project depend totally on the manager’s personal approach and the programmers’ experience, meaning the success of a particular project can be repeated only if the same managers and programmers are assigned to the next project. In addition, if managers or programmers leave the company, the quality of produced software will sharply decrease. In many cases, the development process comes down to writing code with minimal testing.
2. **Repeatable level.** At this level, project management technologies have been introduced in a company. That project planning and management is based on accumulated experience and there are standards for produced software (these standards are documented) and there is a special quality management group. At critical times, the process tends to roll back to the initial level.
3. **Defined level.** Here, standards for the processes of software development and maintenance are introduced and documented (including project management). During the introduction of standards, a transition to more effective technologies occurs. There is a special quality management department for building and maintaining these standards. A program of constant, advanced training of staff is required for achievement of this level. Starting with this level, the degree of organizational dependence on the qualities of particular developers decreases and the process does not tend to roll back to the previous level in critical situations.
4. **Managed level.** There are quantitative indices (for both software and process as a whole) established in the organization. Better project management is achieved due to the decrease of digression in different project indices. However, sensible variations in process efficiency may be different from random variations (noise), especially in mastered areas.
5. **Optimizing level.** Improvement procedures are carried out not only for existing processes, but also for evaluation of the efficiency of newly introduced innovative technologies. The main goal of an organization on this level is permanent improvement of existing processes. This should anticipate possible errors and defects and decrease the costs of software development, by creating reusable components for example.

A 37 :

* It is a type of non-functional testing.
* Security testing is basically a type of [software testing](http://istqbexamcertification.com/what-is-a-software-testing/) that’s done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
* It is a process to determine that an information system protects data and maintains functionality as intended.
* The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software’s and hardware’s and firewall etc.
* Software security is about making software behave in the presence of a malicious attack.
* The six basic security concepts that need to be covered by security testing are: confidentiality, integrity, authentication, availability, authorization and non-repudiation.

Types of Security testing :

* **Vulnerability Scanning**: This is done through automated software to scan a system against known vulnerability signatures.
* **Security Scanning:** It involves identifying network and system weaknesses, and later provides solutions for reducing these risks. This scanning can be performed for both Manual and Automated scanning.
* **Penetration testing**: This kind of testing simulates an attack from malicious hacker. This testing involves analysis of a particular system to check for potential vulnerabilities to an external hacking attempt.
* **Risk Assessment:**This testing involves analysis of security risks observed in the organization. Risks are classified as  Low, Medium and High. This testing recommends controls and measures to reduce the risk.
* **Security Auditing:** This is internal inspection of Applications and Operating systems for security flaws. Audit can also be done via line by line inspection of code
* **Ethical hacking:** It's hacking an Organization Software systems. Unlike malicious hackers ,who steal for their own gains , the intent is to expose security flaws in  the system.
* **Posture Assessment:** This combines Security scanning, Ethical Hacking and Risk Assessments to show an overall security posture of an organization.

A 39 :

**What is Load Testing?** *How fast you can drive your car.*

Load Testing help us to study the behavior of the application under various loads. The main parameter to focus is response time. This study reveals how many concurrent users that server can handle effectively and quickly.

**What is Stress Testing?** *You can drive the car fast, but what is the speed limit*

Stress Testing help us to observe the stability of the application. The main intention is to identify the breaking point of the server. The main parameter to focus is throughput (how much data is being transferred between client and server)

**Differences between Load Testing and Stress Testing**

|  |  |
| --- | --- |
| **Load Testing** | **Stress Testing** |
| Helps us to determine the reliability of the application. | Help us to observe the stability of the application. |
| Help us to study the behavior of the application under various loads | The main intention is to identify the breaking point of the server |
| The main parameter to focus is response time. | The main parameter to focus is throughput. |
| Load Testing is the subset of stress testing. | Stress Testing is the super set of Load Testing. |

Q 41 :

**What is Regression Testing?**  
Regression means retesting the unchanged parts of the application. Test cases are re-executed in order to check whether previous functionality of application is working fine and new changes have not introduced any new bugs. This test can be performed on a new build when there is significant change in original functionality or even a single bug fix.

This is the method of verification. Verifying that the bugs are fixed and the newly added features have not created in problem in previous working version of software.