|  |
| --- |
| Comsats University Logo 1 - UpLabs |
| SOFTWERE TESTING |
| MUHAMMAD SHERAZ |
| **FA19-bse-160**  **Section :6th\_D** |
| **6/22/2022** |

**QUESTION NO #01**

**1. How can we conduct**

* **Static code analyses**
* **Mutation testing**
* **Data flow testing**

**ANSWER:**

1. **Static Code Analysis**

**Definition:**

“Static code analysis is a method of debugging by examining source code before a program is run. It’s done by analyzing a set of code against a set (or multiple sets) of coding rules.”

**Explanation:**

Static code analysis and static analysis are often used interchangeably, along with source code analysis. This type of analysis addresses weaknesses in source code that might lead to vulnerabilities. Of course, this may also be achieved through manual code reviews. But using automated tools is much more effective.

**Programming Language**

Analyzers are designed for many different programming languages. So, it’s important to choose a tool that supports your language.

**Standards**

One the primary uses of static analyzers is to comply with standards. So, if you’re in a regulated industry that requires a [coding standard](https://www.perforce.com/resources/qac/coding-standards), you’ll want to make sure your tool supports that standard.

1. **Static Code Analysis**

**Definition:**

Mutation testing is a white box method in software testing where we insert errors purposely into a program (under test) to verify whether the existing test case can detect the error or not. In this testing, the mutant of the program is created by making some modifications to the original program.

**Explanation:**

The primary objective of mutation testing is to check whether each mutant created an output, which means that it is different from the output of the original program. We will make slight modifications in the mutant program because if we change it on a massive scale than it will affect the overall plan.

When we detected the number of errors, it implies that either the program is correct or the test case is inefficient to identify the fault.

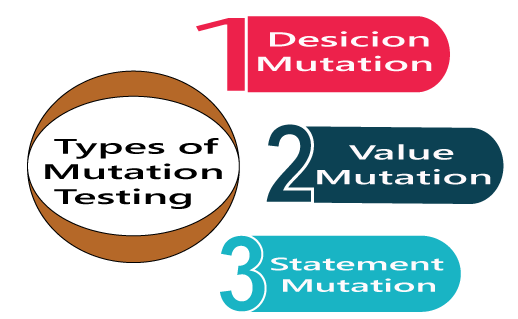
Mutation testing purposes is to evaluate the quality of the case that should be able to fail the mutant code hence this method is also known as Fault-based testing as it used to produce an error in the program and that why we can say that the mutation testing is performed to check the efficiency of the test cases.

**Types of mutation testing**

Mutation testing can be classified into three parts, which are as follows:

* Decision mutations
* value mutations
* Statement mutation

**Diagram:**



1. Decision mutations

In this type of mutation testing, we will check the design errors. And here, we will do the modification in arithmetic and logical operator to detect the errors in the program.

Like if we do the following changes in arithmetic operators:

* plus(+)→ minus(-)
* asterisk(\*)→ double asterisk(\*\*)
* plus(+)→incremental operator(i++)

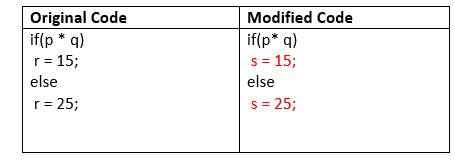
1. Value mutations

In this, the values will modify to identify the errors in the program, and generally, we will change the following:

* Small value à higher value
* Higher value à Small value.

1. Statement Mutations

Statement mutations means that we can do the modifications into the statements by removing or replacing the line as we see in the below example:



1. **DATA Flow Testing**

**Definition:**

Data flow testing is used to analyze the flow of data in the program. It is the process of collecting information about how the variables flow the data in the program. It tries to obtain particular information of each particular point in the process.

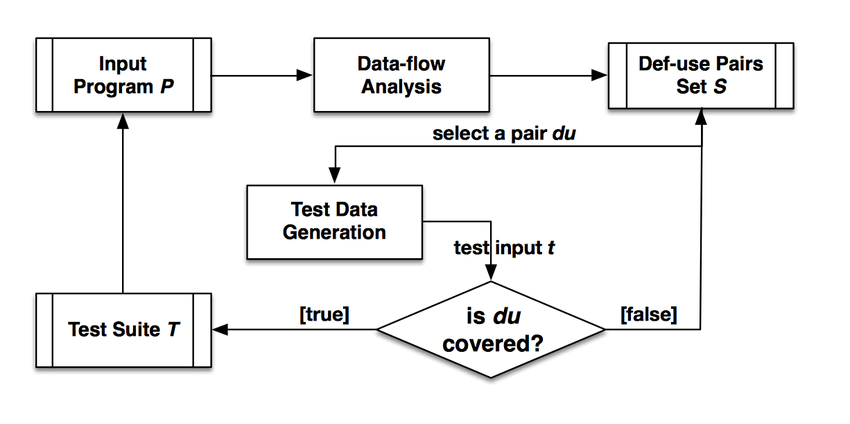
**Explanation:**

Data flow testing is a group of testing strategies to examine the control flow of programs in order to explore the sequence of variables according to the sequence of events. It mainly focuses on the points at which values assigned to the variables and the point at which these values are used by concentrating on both points, data flow can be tested.

Data flow testing uses the control flow graph to detect illogical things that can interrupt the flow of data. Anomalies in the flow of data are detected at the time of associations between values and variables due to:

* If the variables are used without initialization.
* If the initialized variables are not used at least once.

**Diagram:**

****

**THE END**