1.1 Humans, errors and testing

Contents

Errors

Errors are a part of our daily life.

Humans make errors in their thoughts, actions, and in the products that might result from their

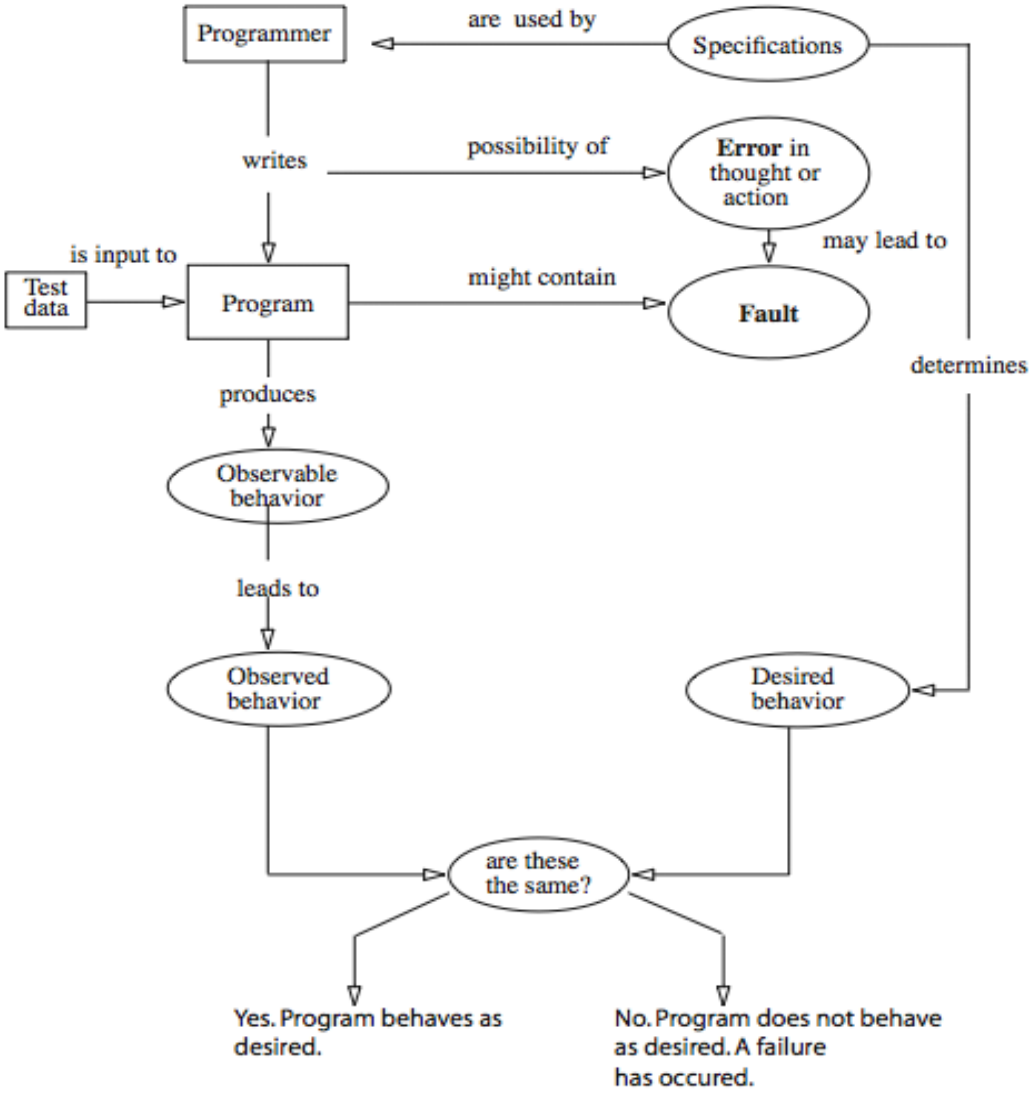
actions.

Errors occur wherever humans are involved in taking actions and making decisions.

These fundamental facts of human existence make testing an essential activity.

Contents

Error, faults, failures



Contents

1.2 Software Quality

Contents

Software quality

Static quality attributes: structured, maintainable, testable code as well as the availability of correct and complete documentation.

Dynamic quality attributes: software reliability, correctness, completeness, consistency, usability, and performance

Contents

Software quality (contd.)

Completeness refers to the availability of all features listed in the requirements, or in the user manual. An incomplete software is one that does not fully implement all features required.

Consistency refers to adherence to a common set of conventions and assumptions. For example, all buttons in the user interface might follow a common color coding convention. An example of inconsistency would be when a database application displays the date of birth of a person in the database.

Contents

Software quality (contd.)

Usability refers to the ease with which an application can be used. This is an area in itself and there exist techniques for usability testing. Psychology plays an important role in the design of techniques for usability testing.

Performance refers to the time the application takes to perform a requested task. It is considered as a *non-functional requirement*. It is specified in terms such as ``This task must be performed at the rate of X units of activity in one second on a machine running at speed Y, having Z gigabytes of memory."

Contents

1.3 Requirements, behavior, and correctness

Contents

Requirements, behavior, correctness

Requirements leading to two different programs:

Requirement 1: It is required to write a

program that inputs two integers and outputs the maximum of these.

Requirement 2: It is required to write a

program that inputs a sequence of integers and outputs the sorted version of this sequence.

Contents

Input domain (Input space)

*The set of all possible inputs to a program **P** is known as the input domain or input space, of **P**.*

Using Requirement 1 above we find the input domain of **max**

to be the set of all pairs of integers where each element in the pair integers is in the range -32,768 till 32,767.

Using Requirement 2 it is not possible to find the input domain for the sort program.

Contents

Input domain (Continued)

Modified Requirement 2:

It is required to write a program that inputs a

sequence of integers and outputs the integers in this sequence sorted in either ascending or descending order. The order of the output sequence is determined by an input request character which should be ``A" when an ascending sequence is desired, and ``D" otherwise.

While providing input to the program, the request character is input first followed by the sequence of integers to be sorted; the sequence is terminated with a period.

Contents

Input domain (Continued)

Based on the above modified requirement, the input domain for **sort** is a set of pairs. The first element of the pair is a character. The second element of the pair is a sequence of zero or more integers ending with a period.

Contents

Valid/Invalid Inputs

The modified requirement for sort mentions that the

request characters can be ``A" and ``D", but fails to answer the question ``What if the user types a different character ?''

When using **sort** it is certainly possible for the user to type a

character other than ``A" and ``D". Any character other than ``A'' and ``D" is considered as invalid input to **sort**. The requirement for **sort** does not specify what action it should take when an invalid input is encountered.

Contents

1.4 Correctness versus reliability

Contents

Correctness

Though correctness of a program is desirable, it is almost never the objective of testing.

To establish correctness via testing would imply testing a program on all elements in the input domain. In most cases that are encountered in practice, this is impossible to accomplish.

Thus, correctness is established via mathematical proofs of programs.

Contents

Correctness and Testing

While correctness attempts to establish that the program is error free, testing attempts to find if there are any errors in it.

Thus, completeness of testing does not necessarily demonstrate that a program is error free.

Testing, debugging, and the error removal processes together increase our confidence in the correct functioning of the program under test.

Contents

Software reliability: two definitions

Software reliability [ANSI/IEEE Std 729-1983]: is the probability of failure free operation of software over a given time interval and under given conditions.

Software reliability is the probability of failure free operation of software in its intended environment.

Contents

1.5 Testing and debugging

Contents

Testing and debugging

Testing is the process of determining if a program has any errors.

When testing reveals an error, the process used to determine the cause of this error and to remove it, is known as debugging.

Contents

Testing and verification

Program verification aims at proving the correctness of programs by showing that it contains no errors. This is very different from testing that aims at uncovering errors in a program.

Program verification and testing are best considered as complementary techniques. In practice, program verification is often avoided, and the focus is on testing.

Contents

Testing and verification (contd.)

Testing is not a perfect technique in that a program might contain errors despite the success of a set of tests.

Verification promises to verify that a program is free from errors. However, the person/tool who verified a program might have made a mistake in the verification process; there might be an incorrect assumption on the input conditions; incorrect assumptions might be made regarding the components that interface with the program, and so on.

Verified and published programs have been shown to be incorrect.

Contents

Summary

We have dealt with some of the most basic concepts in software testing..

END OF LECTURE

Contents