Chapter 1

Basics of Software Testing

The purpose of this introductory chapter is to familiarize the reader with the basic concepts related to software testing. In this chapter we will set up a framework for the remainder of this book. Specific questions, answered in substantial detail in subsequent chapters, will be raised here. After reading this chapter, you are likely to be able to ask meaningful questions related to software testing and reliability.

1.1. Humans, errors, and testing

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 almost everywhere. For example, humans make errors in speech, in medical prescription, in surgery, in driving, in observation, in sports, and, certainly in love and software development. Table 1.1 provides examples of human errors. The consequences of human errors vary significantly. An error might be insignificant in that it leads to a gentle friendly smile, such as when a slip of tongue occurs. Or, an error may lead to a catastrophe, such as when an operator fails to recognize that a relief valve on the pressurizer was stuck open and this resulted in a disastrous radiation leak.

To determine whether or not there are any errors in our thought, actions, and the products generated, we resort to the process of *testing*. The primary goal of testing is to determine if the thoughts, actions, and products are as desired, i.e. they conform to the requirements. Testing of thoughts is usually designed to determine if a concept or method has been understood satisfactorily. Testing of actions is designed to check if a skill that results in the actions has been acquired satisfactorily. Testing of a product is designed to check if the product behaves as desired. Note that both syntax and semantic errors arise during programming. Given that most modern compilers are able to detect syntactic errors, testing focuses on semantic errors, also known as faults, that cause the program under test to behave incorrectly.

EXAMPLE 1.1. An instructor administers a test to determine how well the students have un-

Table 1.1: Examples of errors in various fields of human endeavor.

Area	Error
Hearing	Spoken: He has a garage for repairing *foreign* cars. Heard: He has a garage for repairing *falling* cars.
Medicine	Incorrect antibiotic prescribed.
Music performance	Incorrect note played.
Numerical analysis	Incorrect algorithm for matrix inversion.
Observation	Operator fails to recognize that a relief valve is stuck open.
Software	Operator used: $\neq$, correct operator: $>$. Identifier used: new_line, correct identifier: next_line. Expression used: $a \wedge (b \vee c)$, correct expression: $(a \wedge b) \vee c$. Data conversion from 64-bit floating point to 16-bit integer not protected (resulting in a software exception).
Speech	Spoken: *waple malnut*, intent: *maple walnut*. Spoken: *We need a new refrigerator*, intent: *We need a new washing machine*.
Sports	Incorrect call by the referee in a tennis match.
Writing	Written: What kind of *pans* did you use ? Intent: What kind of *pants* did you use ?

derstood what the instructor wanted to convey. A tennis coach administers a test to determine how well the understudy makes a serve. A software developer tests the program developed to determine if it behaves as desired. In each of these three cases there is an attempt by a tester to determine if the human thoughts, actions, and products behave as desired. Behavior that deviates from the desirable is possibly due to an error. ■

EXAMPLE 1.2. "Deviation from the expected" may *not* be due to an error for one or more reasons. Suppose that a tester wants to test a program to sort a sequence of integers. The program can sort an input sequence in both descending or ascending orders depending on the request made. Now suppose that the tester wants to check if the program sorts an input sequence in *ascending* order. To do so, he types in an input sequence and a request to sort the sequence in *descending* order. Suppose that the program is correct and produces an output which is the input sequence in descending order.

Upon examination of the output from the program, the tester hypothesises that the sorting program is incorrect. This is a situation where the tester made a mistake (an error) that led to his incorrect interpretation (perception) of the behavior of the program (the product). ■

1.1.1. Errors, faults, and failures

There is no widely accepted and precise definition of the term "error." Figure 1.1 illustrates one class of meanings for the terms error, fault, and failure. A programmer writes a program. An *error* occurs in the process of writing a program. A *fault* is the manifestation of one or more errors. A failure occurs when a faulty piece of code is executed leading to an incorrect state that propagates to the program's output. The programmer might misinterpret the requirements and consequently write incorrect code. Upon execution, the program might display behavior that does not match with the expected behavior implying thereby that a *failure* has ocurred. A fault in the program is also commonly referred to as a *bug* or a *defect*. The terms error and bug are by far the most common ways of referring to something "wrong" in the program text that might lead to a failure. In this text we often use the terms "error" and "fault" as synonyms. Faults are sometimes referred to as *defects*.

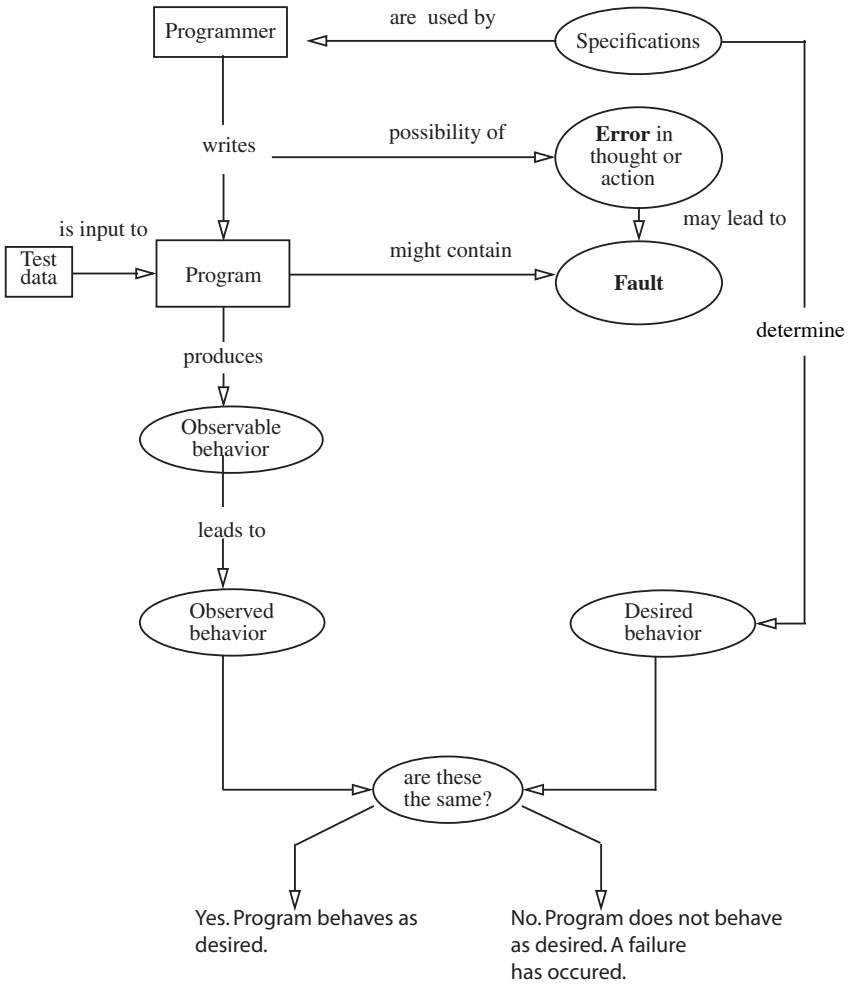

Figure 1.1: Errors, faults, and failures in the process of programming and testing.

In Figure 1.1 notice the separation of "observable" from "observed" behavior. This sepa-