Auditoría, Calidad y Gestión de Sistemas
(ACG)

Práctica
Errors, Faults y Failures

Curso 22/23

0.1 Objetivos de aprendizaje

- Conocer la diferencia entre errors, faults y failures
- Darse cuenta que podemos tener tests que llegan al estado de error, ejecutando faults SIN ver un failure

0.2 Tareas

1 Leer el apéndice de este documento (en inglés) para entender la diferencia entre errors, faults y failures.

2 Bajar los .java de poliformat de unos programas muy simles (findLast, lastZero, countPositive, oddOrPos y sus tests).

3 Analizar los tests JUnit.

4 Para cada programa:
   a Reflexionar, Qué está mal implementado en el còdigo? Cuál es el fault y cómo se puede arreglar?
   b Si es posible, implementa un caso de test en JUnit que **no** ejecuta el fault.
   c Si es posible, implementa un caso de test en JUnit que ejecuta el fault, pero **no** resulta en failure.
   d Si es posible, implementa un caso de test en JUnit que ejecuta el fault, y **sí** resulta en failure.
   e Implementar una solución al fault y testear bien que el código ahora sea correcto.

APPENDIX: Terminology on errors, faults and failures

In testing, there is no consensus on the terminology for the concepts of error, fault, failure, incidents and bugs. For this course, we will stick to the terminology used by ISTQB, the International Software Testing Qualification Board[1], as described in [2]. But do not assume that everybody uses this. Therefore when you discuss a problem with someone check that you have the same understanding of the terminology.

DEFINITION .1 *Error* A human action that produces an incorrect result, for example a mistake, a misunderstanding, a misconception, etc.

DEFINITION .2 *Fault or defect* A flaw in a component or system (e.g. an incorrect statement or data definition) that can cause the component or system to enter an incorrect state (e.g. variable gets assigned the wrong value). A fault, if encountered during execution, may cause a failure of the component or system but it can also go unnoticed.

DEFINITION .3 *Failure* A deviation of the component or system from its expected delivery, service or result.

To gain a better understanding of the relation between the three concepts we will adapt an analogy from [1].

Consider a medical doctor diagnosing a patient. The patient informs the doctor with a list of symptoms (*failures*). The doctor must then discover the *error*, that is the root cause of the symptoms. To help make the diagnosis, the doctor might want to do some tests that look for anomalous internal conditions such as high blood pressure, irregular heartbeat, high cholesterol, et cetera. These conditions would correspond to the *faults*.

For examples of errors, faults and failures in software, let us look at a code example also borrowed from [1].

```
/**
 * Counts the amount of zeros in an array
 * @param x: the array to count zeros in
 * @return: the number of occurrences of 0 in array x
 * @throws: NullPointerException if array x is null
 */
public static int numZero (int[] x) {
  int count = 0;
  for (int i=1; i < x.length; i++) {
    if (x[i] == 0) count++;
  }
  return count;
}
```

Error
The programmer has made a mistake writing this code. Maybe the programmer made a typo: typing 1 instead of 0. Maybe the programmer did not know that in Java the first element of an array resides at index 0. Maybe the programmer re-used some code and forgot to adjust the index.

[1] http://www.istqb.org/

Fault
As a result of this error, the first element in the array is never checked to be zero and so is not counted if it happens to be zero.

Failure
The fault only propagates to a failure that is visible to the user when `numZero` is called with an array that has a zero in the first element:

input	`[0, 4, 6, 8]`
expected result	1
actual result	0
verdict	FAILURE

If there is no zero in the first element, the fault will be executed but does not result in a failure.

input	`[1, 4, 0, 8]`
expected result	1
actual result	1
verdict	CORRECT

It is clear what a failure is: some incorrect behaviour of the software that is visible to the user. However, the distinction between error and fault might be more obscure. Therefore, we do not want to spend too much time on distinguishing errors and faults. Most of the time these two definitions can be used interchangeably anyway.

We end this section on terminology by explaining a few more words used in this context: bug, issue and incident. Perhaps you have already discovered on the Internet that the software engineering community has not yet reached a consensus on which words to use for what.

Bug
If you look up the word *bug* on the Internet, you will find definitions that include all the words we have defined above. Some will define it as an error, others will define it as a fault, and yet other definitions relate bugs only to failures. On Wikipedia[2] they play safe by mentioning them all:

A software bug is an error, flaw, defect, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

Also, you will find many people having all kinds of opinions about whether the word bug may still be used. We will not go into that here in this course.

Incident
The word *incident* is often used when something suspicious has happened, but it is not yet clear what the failure is. It is a symptom that something is wrong and that alerts the tester or user that a failure might come.

[2]https://en.wikipedia.org/wiki/Software_bug

Issue
The word *issue* is used in an even broader sense to state that something is going on but without making claims about where it comes from, if it is a failure due to some fault, or whether it should be fixed. This terminology is sometimes used such that the customer cannot claim that things should be fixed during the warranty period of a software program since they are not recognised as real failures. Also, the word is sometimes used to avoid offending programmers and prevent harm to the team morale.

Bibliography

[1] Paul Ammann and Jeff Offutt. *Introduction to software testing*. Cambridge University Press, 2017.

[2] A. Spillner, T. Linz, and H. Schaefer. *Software Testing Foundations: A Study Guide for the Certified Tester Exam*. Rocky Nook, 2014.