Laboratórios de Informática I
2020/2021

Mestrado Integrado em Engenharia Informática

Universidade do Minho

Sessão Laboratorial - Teste de Software

A Importância do Teste

O teste de software é um componente essencial no desenvolvimento de um produto. Os testes permitem confirmar que o programa:

- faz o que deve fazer (teste funcional);
- tem o desempenho necessário;
- funciona em diferentes plataformas;
- está em conformidade com *standards* de qualidade...

A Importância do Teste

Acima de tudo, o teste serve para encontrar falhas atempadamente e num ambiente controlado.

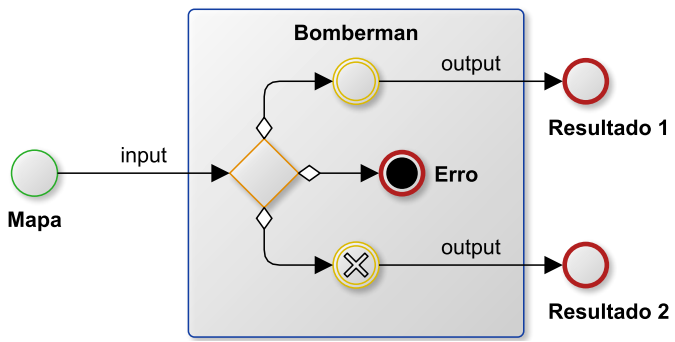
Fonte: http://testsidestory.com/

Métodos de Teste de Software

As falhas podem surgir de várias formas, pelo que também há vários tipos, níveis e estratégias de teste. Por exemplo:

Black-box testing testa sem conhecimento do código.

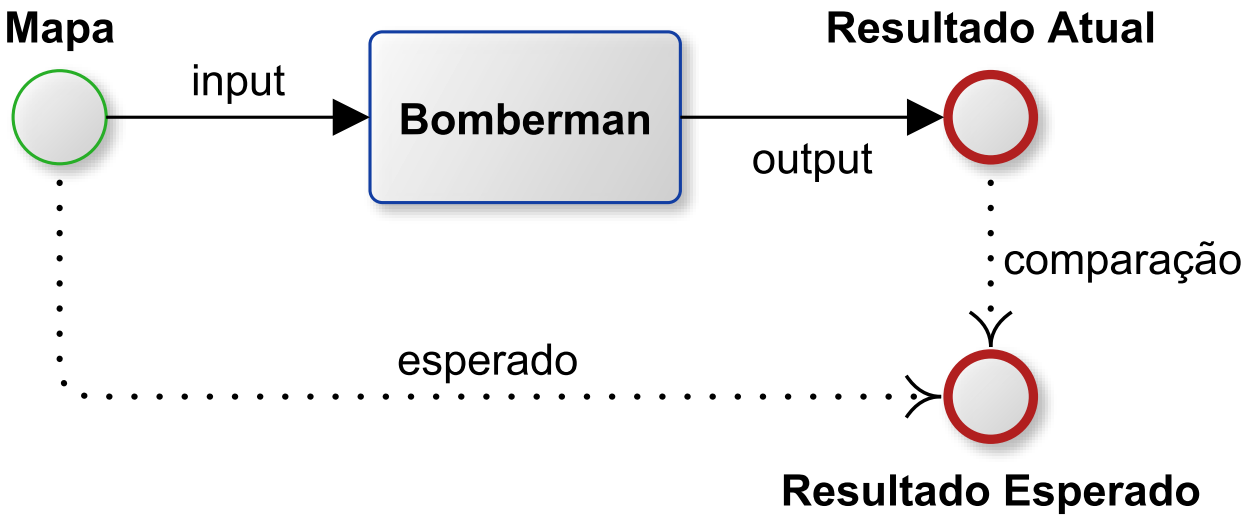
White-box testing testa com conhecimento total do código.



Black-box Testing

No projeto prático abordaremos o black-box testing.

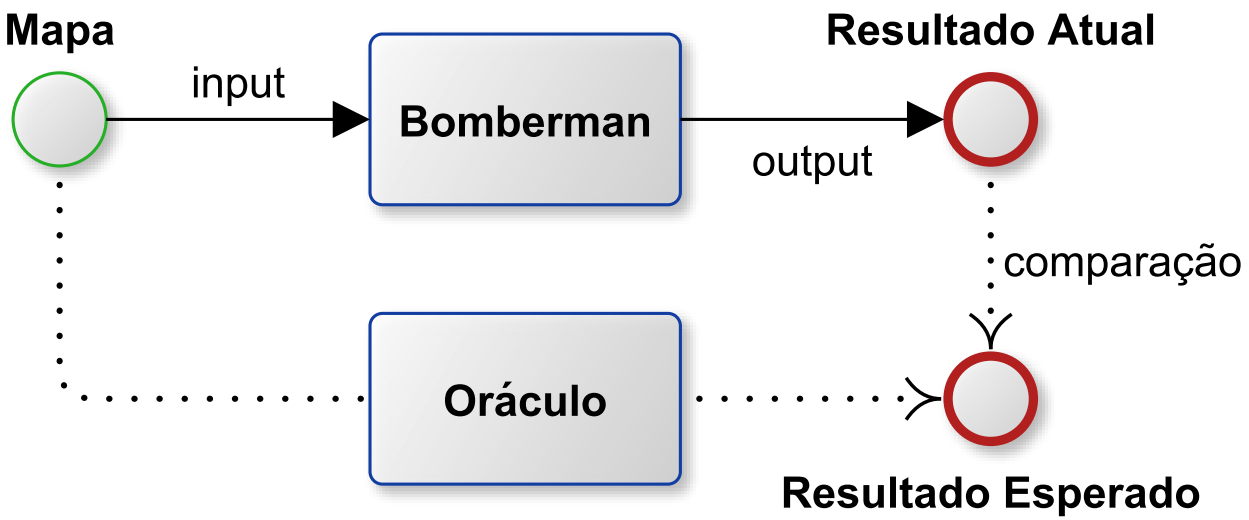
Este tipo de teste consiste em definir uma relação de pares `input-output` e (automaticamente) executar o programa testado, com cada um dos `input`, comparando o `output` do programa ao `output` esperado.



Black-box Testing

Uma técnica possível para gerar estes pares `input-output` é através de um oráculo, uma implementação de referência que estabelece o comportamento esperado do programa testado.

Neste caso é apenas necessário definir os `input` a testar.



Exemplo

```
subLista :: Int -> [a] -> [[a]]
subLista _ [] = []
subLista n l = take n l: subLista n (drop n l)

testSubLista1 = subLista 4 [1..20]
testSubLista2 = subLista 4 [1,2,3,4]
testSubLista3 = subLista 4 [1]
testSubLista4 = subLista 1 [1..20]
testSubLista5 = subLista 15 [1..20]
testSubLista6 = subLista 15 "eu adoro programar em haskell"
```

Outra forma

```
testCasesSubLista :: [(Int, [Int])]
testCasesSubLista = [(4, [1..20]), (4, [1,2,3,4]),
  (4, [1]), (1, [1..20]), (15, [1..20])]
   -- (15, "eu adoro programar em haskell")

testSubLista :: [(Int, [a])] -> [[[a]]]
testSubLista [] = []
testSubLista ((i, l):t) = (subLista i l) : testSubLista t

printResults :: Show a => [a] -> IO ()
printResults m =
  mapM_ (\a -> putStr ("\ntest >>>" ++ show a)) m

*Tarefa1> printResults  $ testSubLista testCasesSubLista
```

Tarefa 1

```
testesT1 :: [(Int, Int, Int)]
```