

Exemplo sobre o Exercício 1 - REO 2: Defeito, Erro e Falha

Considere o programa abaixo:

```
1. public static int numZero (int[] x){
2. //Effects: if x==null throw NullPointerException
3. //else return the number of occurrences of 0 in x
4. int count = 0;
5. for(int i=1; i<x.length;i++){
    a. if(x[i] ==0){
        i. count++;
    b. }
6. }
7. return count;
8. }
```

Defeito do programa:

for(int i =1; i < x.length; i++) deveria ser: for(int i =0; i < x.length; i++).

1. Caso de teste 1:
 - t1 = ([2,7,0], 1) //(vetor x, saída esperado do programa)
 - Para numZero([2,7,0]) a saída será correta, valor 1.
2. Caso de teste 2:
 - t2 = ([0,7,2], 1)
 - Para numZero([0,7,2]) a saída será incorreta, valor 0.

Para os dois casos de teste acima, o **defeito** é executado, um **erro** é apresentado para ambos, pois um estado incorreto é atingido (ou seja, o vetor começa a ser lido da posição 1 ao invés da posição zero), porém somente para o t2 uma **falha** ocorre no programa.

Para entender o conceito de “estado de erro”, é preciso identificar o estado do programa. O estado para o programa numZero corresponde aos valores para as variáveis x, count, i e o contador do programa PC, que significa a linha em que está sendo executada naquele momento (ou o comando em execução). Considerando a execução de t1, o estado do comando if na 1ª iteração do loop é (x = [2,7,0], count = 0, i = 1, PC = if). Observe que este estado está em erro porque o valor de i deveria ser 0 e não 1 na 1ª iteração. Porém, considerando que o valor de count é coincidentemente correto, o estado de erro não se propaga para a saída e, conseqüentemente, o programa não falha. Isso é chamado de **correção coincidente** (falsa impressão de que o programa está correto). Ou seja, um estado do programa está em erro se ele não executa o estado esperado. Para a execução de t2, o correspondente estado é (x = [0,7,2], count=0,i=1, PC=if). Neste caso, o estado de erro aparece na variável count e irá retornar um valor incorreto para o método. Conseqüentemente, uma falha resulta.

Essas definições são importantes pois permitem distinguir a atividade de teste da atividade de depuração. **Teste de software:** avaliar o software, observando sua execução. **Depuração:** processo de encontrar um defeito dado uma falha na execução do software (resultado de uma atividade de teste bem sucedida).