Resumo Teste de caixa branca

- Esse teste se define pela informação que iremos testar obtida através do código fonte, ou melhor da sua lógica.
- O teste de caixa branca está preocupado com o grau em que os casos de teste exercem ou cobrem a lógica (código-fonte) do programa.
- O critério de cobertura da declaração é tão fraco que geralmente é inútil.
- A cobertura lógica mais forte é conhecida como cobertura de decisão ou cobertura de ramificação. Deve-se escrever casos de teste suficientes para que cada decisão tenha um resultado verdadeiro e um resultado falso pelo menos uma vez. Ou seja, cada direção de ramificação deve ser percorrida pelo menos uma vez.
- A cobertura de decisão geralmente pode satisfazer a cobertura de instrução. Como cada instrução está em algum subcaminho que emana de uma instrução de ramificação ou do ponto de entrada do programa, cada instrução deve ser executada se todas as direções de ramificação forem executadas.
- Exceções da cobertura de decisão:
    - Programas sem decisões.
    - Programas ou sub-rotinas/métodos com múltiplos pontos de entrada. Uma determinada instrução pode ser executada apenas se o programa for inserido em um ponto de entrada específico.
    - Declarações dentro das unidades ON. Atravessar todas as direções de ramificação não necessariamente fará com que todas as unidades ON sejam executadas.
- A cobertura de decisão requer que cada decisão tenha um resultado verdadeiro e um falso, e que cada afirmação seja executada pelo menos uma vez.
    - Uma maneira alternativa e mais fácil de expressar isso é que cada decisão tem um resultado verdadeiro e um falso, e que cada ponto de entrada (incluindo unidades ON) seja invocado pelo menos uma vez.
    - Ela considera apenas decisões ou ramificações bidirecionais e deve ser modificada para programas que contêm decisões de vários caminhos.
    - Exemplo: programas Java contendo instruções switch-case, programas Fortran contendo instruções IF aritméticas (três vias) , ou instruções GOTO computadas , ou aritméticas e , programas

COBOL contendo instruções GOTO alteradas ,ou instruções GO-TO DEPENDING-ON.
    - Para esses programas, o critério é exercitar cada resultado possível de todas as decisões pelo menos uma vez e invocar cada ponto de entrada do programa ou sub-rotina pelo menos uma vez.
- A cobertura de decisão é um critério mais forte do que a cobertura de declaração, porém ainda é bastante fraca. Ela tem apenas 50% de chance de explorarmos o caminho onde x não é alterado (ou seja, somente se escolhermos a primeira alternativa). Se a segunda decisão estivesse errada (se deveria ter dito X<1 em vez de X>1), o erro não seria detectado pelos dois casos de teste no exemplo anterior.
- Mais forte do que a cobertura de decisão é a cobertura de condição. Escreva casos de teste suficientes para garantir que cada condição em uma decisão assuma todos os resultados possíveis pelo menos uma vez. Mas, como na cobertura de decisão, isso nem sempre leva à execução de cada instrução, portanto, uma adição ao critério é que cada ponto de entrada do programa ou sub-rotina, bem como as unidades ON, sejam invocados pelo menos uma vez.
- A cobertura de condição geralmente é superior à cobertura de decisão, pois pode (mas nem sempre) fazer com que cada condição individual em uma decisão seja executada com ambos os resultados, enquanto a cobertura de decisão não.
- Cobertura de decisão/condição requer casos de teste suficientes para que cada condição em uma decisão receba todos os resultados possíveis pelo menos uma vez, cada decisão receba todos os resultados possíveis pelo menos uma vez e cada ponto de entrada seja invocado pelo menos uma vez.
- Um ponto fraco da cobertura de decisão/condição é que, embora possa parecer exercer todos os resultados de todas as condições, frequentemente não o faz, porque certas condições mascaram outras condições
- Condição múltipla: Escreva casos de teste suficientes para que todas as combinações possíveis de resultados de condição em cada decisão, e de todos os pontos de entrada, sejam invocados pelo menos uma vez.
- No caso de loops, o número de casos de teste exigidos pelo critério de condição múltipla é normalmente muito menor que o número de caminhos.

- Em resumo, para programas contendo apenas uma condição por decisão, um critério de teste mínimo é um número suficiente de casos de teste para:
    - 1: Invocar todos os resultados de cada decisão pelo menos uma vez e
    - 2: Invocar cada ponto de entrada (como ponto de entrada ou unidade ON) pelo menos uma vez, para garantir que todas as instruções sejam executadas pelo menos uma vez
- Para programas contendo decisões com múltiplas condições, o critério mínimo é um número suficiente de casos de teste para invocar todas as combinações possíveis de resultados de condição em cada decisão e todos os pontos de entrada no programa, pelo menos uma vez.