



Técnicas de teste

Teste caixa branca

Depuração

Teste de ambiente *web*



Técnicas de teste

O objetivo principal do processo de teste de software é detectar a presença de erros no sistema testado. Sendo assim, o teste bem sucedido é aquele que consegue determinar situações nas quais o software falhe. Para se alcançar tal objetivo, diversas são as técnicas que podem ser empregadas. A seguir são descritas algumas dessas técnicas.



Técnicas de teste

O objetivo de uma técnica de teste é ajudar a identificar as condições de teste, os casos de teste e os dados de teste.


Algumas técnicas são mais aplicáveis em determinadas situações e níveis de teste, outras são aplicáveis em todos os níveis de teste. Ao criar casos de teste, os testadores geralmente usam uma combinação de técnicas de teste para obter os melhores resultados do esforço de teste.



Técnicas de teste

A escolha de quais técnicas de teste usar depende de vários fatores, incluindo:

- Ferramentas disponíveis;
- Tempo e orçamento;
- Modelo de ciclo de vida de desenvolvimento de software.
- Os tipos de defeitos esperados no componente ou sistema.



Categorias de técnicas de teste e suas características

A norma ISO/IEC/IEEE 29119-4 (ISO/IEC/IEEE 29119-4, 2015) define três técnicas de teste:

- Técnica baseada em especificação, que é a técnica caixa preta mencionada acima, também conhecida como técnica funcional;
- Técnica baseada em estrutura, que é a técnica caixa branca mencionada acima, também conhecida como técnica estrutural;
- Técnica baseada em experiência, que é a técnica baseada em erros.



Teste caixa preta, o teste funcional

Essa técnica pode ser adotada quando a equipe de teste tem acesso a especificação do sistema. Isso porque a base para definição dos casos de teste é a especificação do sistema, visto que ao usar essa técnica, o intuito é testar o sistema do ponto de vista das suas funcionalidades.



Teste caixa preta, o teste funcional

Como o próprio nome indica, essa técnica de teste considera que o sistema a ser testado é uma caixa preta – você não sabe o que tem dentro, só sabe o que precisa colocar dentro da caixa e o que você espera que saia da caixa.



Teste caixa preta, o teste funcional

Em outras palavras, trazendo um pouco mais para a realidade do teste, você sabe os dados que devem colocar dentro da caixa, considerando a especificação das funcionalidades que devem estar implementadas no software, e você também sabe qual a saída de dados esperada.

Teste caixa preta, o teste funcional





Teste caixa preta, o teste funcional

De acordo com Myers (Myers, 2004), na técnica funcional o objetivo do testador é não se preocupar com o comportamento interno da estrutura do programa, ao invés disso, o testador deve se concentrar em encontrar situações nas quais o sistema não se comporta da forma como foi especificado.



Teste caixa preta, o teste funcional

Para aplicar esse critério de teste, precisamos dividir o domínio de entrada de dados em classes e partições de equivalência. Classes são divididas em válidas e inválidas e dependendo do requisito e dos dados de entrada, as partições de equivalência são definidas – partições de equivalência é quando qualquer valor, dentro de um dado intervalo, tem a mesma importância.



Teste caixa preta, o teste funcional

De acordo com Maldonado e Fabbri (Maldonado & Fabbri, 2001), os critérios de teste servem para ajudar a definir, selecionar ou revisar os casos de teste a fim de aumentar as possibilidades de identificação de defeitos e estabelecer um nível elevado de confiança dos testes.



Teste caixa preta, o teste funcional

Cada critério possui seus requisitos de teste, que ajudam a gerar os casos de teste de forma a satisfazer o critério escolhido e também avaliar a qualidade de um conjunto de teste existente, analisando quais casos de teste precisam ser acrescentados ao conjunto para que determinado critério seja atingido.



Teste caixa preta, o teste funcional

Os principais critérios de teste caixa preta são:

- Particionamento de Equivalência
- Análise do Valor limite.
- Teste de tabela de decisão
- Teste de transição de estado



Teste caixa preta, o teste funcional

Os principais critérios de teste caixa preta são:

- **Particionamento de Equivalência**
- Análise do Valor limite.
- Teste de tabela de decisão
- Teste de transição de estado



Teste caixa preta, o teste funcional

Particionamento de Equivalência

Para aplicar esse critério de teste, precisamos dividir o domínio de entrada de dados em classes e partições de equivalência. Classes são divididas em válidas e inválidas e dependendo do requisito e dos dados de entrada, as partições de equivalência são definidas – partições de equivalência é quando qualquer valor, dentro de um dado intervalo, tem a mesma importância.



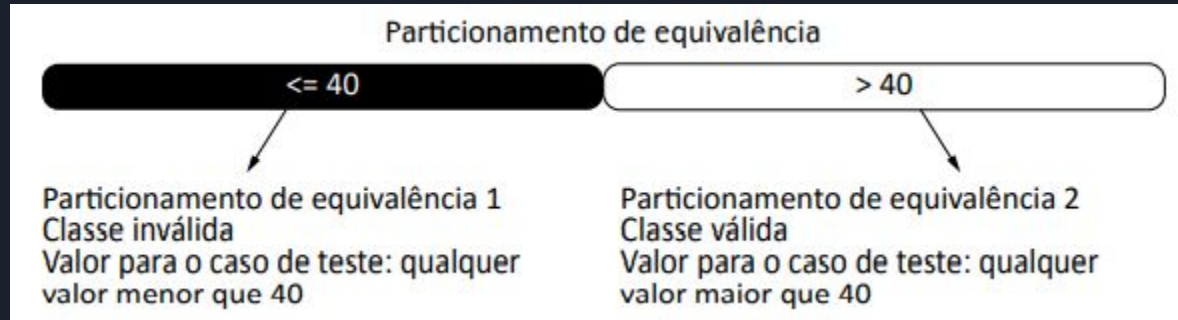
Teste caixa preta, o teste funcional

Particionamento de Equivalência

Se a condição de entrada mencionada no requisito especifica um intervalo (por exemplo, entre 0 e 100) ou um valor específico (número real), devemos definir uma classe válida e duas inválidas; se a condição de entrada mencionada no requisito especifica um membro de um conjunto (por exemplo, um valor a ser selecionado a partir de uma lista, valor maior que 40) ou uma condição booleana (sim/não, verdadeiro/ falso etc.) devemos definir uma classe válida e uma inválida (Fabbri, 2009).

Teste caixa preta, o teste funcional

Particionamento de Equivalência





Teste caixa preta, o teste funcional

Os principais critérios de teste caixa preta são:

- Particionamento de Equivalência
- Análise do Valor limite.
- Teste de tabela de decisão
- Teste de transição de estado



Teste caixa preta, o teste funcional

Análise do valor limite:

O critério de análise do valor limite é complementar ao Particionamento de Equivalência. O foco desse critério é identificar fontes favoráveis a terem defeitos e criar casos de teste para exercitar esses dados (Fabbri, 2009).



Teste caixa preta, o teste funcional

Análise do valor limite:

Casos de teste bem elaborados são feitos para identificar defeitos e, se nesse cenário o testador usar o critério de análise do valor limite, possivelmente o defeito sobre a interpretação do valor 40 seria identificado (se houvesse algum).



Teste caixa preta, o teste funcional

Análise do valor limite:

O requisito do critério análise do valor limite é definir casos de teste para valores de entrada que estão no limite das classes de equivalência. Algo importante de se verificar é que o tipo de dados deve ser observado com atenção.



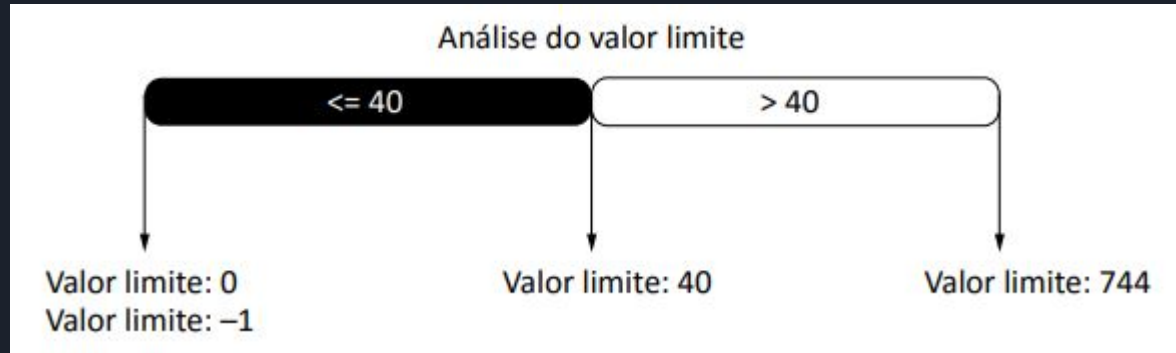
Teste caixa preta, o teste funcional

Análise do valor limite:

Sendo assim, considerando o exemplo anterior, nós devemos pensar em limites e dados propícios a darem problema (ou seja, não estão sendo considerados da forma correta na implementação) – esses serão os valores de entrada para os casos de teste.

Teste caixa preta, o teste funcional

Análise do valor limite:





Teste caixa preta, o teste funcional

Análise do valor limite:

- O valor 40 – que está entre as classes de equivalência;
- Os valores 0 e -1 – que, apesar de não estar especificado, entendemos que 0 e valores negativos podem não se comportar como qualquer valor $0 \leftarrow 40$ e por isso, valores negativos e 0 sempre são usados como limites quando não há especificação clara sobre eles;
- O valor 744 – que é 31×24 , ou seja, nenhum funcionário conseguiria trabalhar mais que 24 horas por dia nos 31 dias do mês. Como não há uma limitação dos requisitos em relação às entradas, um bom testador deve considerar o domínio da aplicação, o requisito em si e pensar em valores que podem resultar em problema



Teste caixa preta, o teste funcional

Os principais critérios de teste caixa preta são:

- Particionamento de Equivalência
- Análise do Valor limite.
- Teste de transição de estado
- Teste de caso de uso



Teste caixa preta, o teste funcional

Teste de transição de estado :

Componentes ou sistemas podem responder de maneira diferente a um evento, dependendo das condições atuais ou do histórico anterior (p. ex., os eventos que ocorreram desde que o sistema foi inicializado).



Teste caixa preta, o teste funcional

Teste de transição de estado :

O histórico anterior pode ser resumido usando o conceito de estados. Um diagrama de transição de estado mostra os possíveis estados do software, bem como a forma como o software entra, sai e transita entre os estados.



Teste caixa preta, o teste funcional

Teste de transição de estado :

Uma transição é iniciada por um evento (p. ex., entrada do usuário de um valor em um campo). O evento resulta em uma transição. Se o mesmo evento pode resultar em duas ou mais transições diferentes do mesmo estado, esse evento pode ser qualificado por uma condição de proteção. A mudança de estado pode fazer com que o software execute uma ação (p. ex., gerar uma mensagem de cálculo ou de erro)



Teste caixa preta, o teste funcional

Os principais critérios de teste caixa preta são:

- Particionamento de Equivalência
- Análise do Valor limite.
- Teste de transição de estado
- Teste de caso de uso



Teste caixa preta, o teste funcional

Teste de caso de uso:

Os testes podem ser derivados de casos de uso, que são uma maneira específica de projetar interações com itens de software, incorporando requisitos para as funções de software representadas pelos casos de uso.



Teste caixa preta, o teste funcional

Teste de caso de uso:

Os casos de uso estão associados a atores (usuários humanos, hardware externo ou outros componentes ou sistemas) e assuntos (o componente ou sistema ao qual o caso de uso é aplicado).



Teste caixa preta, o teste funcional

Teste de caso de uso:

Os casos de uso estão associados a atores (usuários humanos, hardware externo ou outros componentes ou sistemas) e assuntos (o componente ou sistema ao qual o caso de uso é aplicado).



Teste caixa preta, o teste funcional

Teste de caso de uso:

Cada caso de uso especifica algum comportamento que um assunto pode realizar em colaboração com um ou mais atores (UML 2.5.1 2017). Um caso de uso pode ser descrito por interações e atividades, bem como condições prévias, pós-condições e linguagem natural, quando apropriado. Interações entre os atores e o sujeito podem resultar em mudanças no estado do sujeito.



Teste caixa preta, o teste funcional

Sendo assim, com mais quatro casos de teste nós conseguimos a conformidade com os requisitos de outro critério de teste funcional. Esses dois critérios são extremamente úteis e devem fazer parte da mentalidade de qualquer testador e qualquer desenvolvedor, pois faz com que todos os artefatos, do requisito até o código fonte, sejam desenvolvidos de forma a evitar que enganos sejam cometidos no tratamento desses limites.



Teste caixa branca, o teste estrutural

Essa técnica pode ser adotada quando a equipe de teste tem acesso ao código fonte do sistema. Obviamente que ter acesso à especificação do sistema também é desejável, mas é obrigatório o acesso ao código fonte.



Teste caixa branca, o teste estrutural

Essa exigência é devido ao fato que a base para definição dos casos de teste é o código fonte, visto que ao usar essa técnica, o intuito é testar o sistema do ponto de vista da implementação, ou seja, das linhas de código que foram produzidas com o intuito de prover as funcionalidades esperadas do sistema.



Teste caixa branca, o teste estrutural

Como o próprio nome indica, essa técnica de teste considera que o sistema a ser testado é uma caixa branca – você sabe exatamente o que tem dentro da caixa.



Teste caixa branca, o teste estrutural

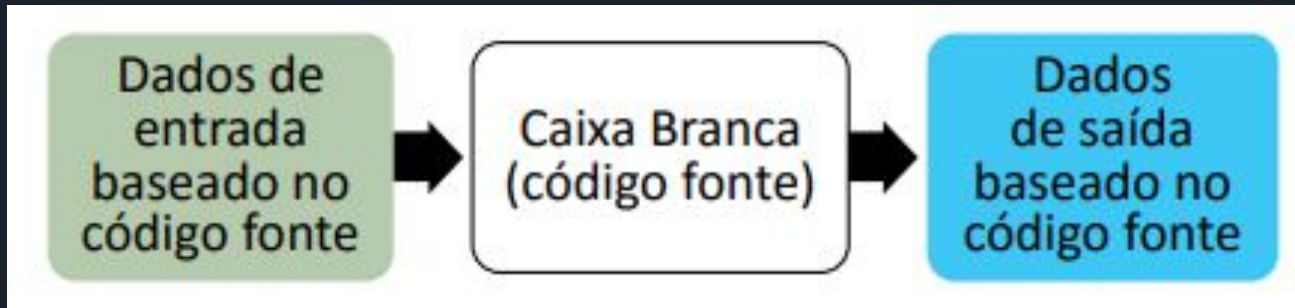
Você continua tendo que saber o que precisa colocar dentro da caixa e o que você espera que saia da caixa, como fazemos no teste caixa preta, mas a grande diferença aqui é que você define “o que colocar dentro da caixa” com base no que tem dentro da caixa.



Teste caixa branca, o teste estrutural

Em outras palavras, trazendo um pouco mais para a realidade do teste, você sabe os dados que deve colocar dentro da caixa e sabe qual a saída de dados esperada, mas nessa técnica, você decide os dados a serem inseridos na caixa com base no código fonte do sistema.

Teste caixa branca, o teste estrutural





Teste caixa branca, o teste estrutural

De acordo com Myers (Myers, 2004), na técnica estrutural o objetivo do testador é se preocupar com o comportamento de cada uma das estruturas internas do programa. Em outras palavras, a preocupação é com a lógica que foi implementada. Quando não há especificação do sistema, essa pode ser uma das formas possíveis de testar o que foi implementado.



Teste caixa branca, o teste estrutural

Voltando no mesmo exemplo que usamos na seção anterior, podemos considerar que o código-fonte (aqui apresentado em pseudocódigo) para implementar o método das horas extras.

Teste caixa branca, o teste estrutural

“Como RH, eu quero informar o número de horas trabalhadas e saber o percentual de horas extras que o funcionário fez, considerando que o número de horas padrão é 40 horas mensais, para que eu saiba se existe um funcionário que faz mais que 25% de hora extra por mês”.

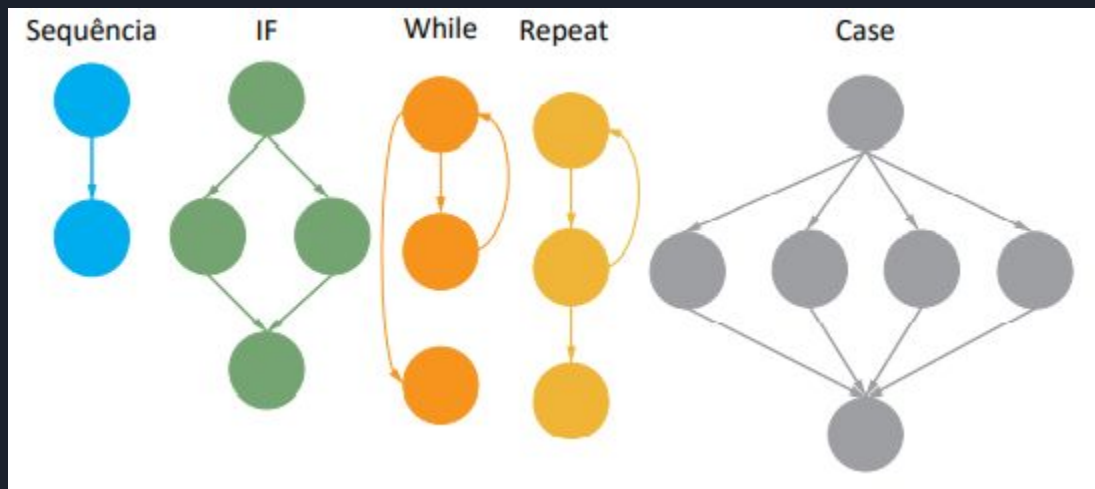
```
1 inteiro calculo_hextraPercentual(real horasTrabalhadas) {
2     inteiro horasMes, horasExtras;
3     horasMes = 40;
4     SE (horasExtras > 40){
5         horasExtras = horasTrabalhadas - 40;
6         return (horasExtras * 100)/horasMes;
7     SENÃO
8         return 0;
9     }
10.}
```



Teste caixa branca, o teste estrutural

Para facilitar a definição dos casos de teste da técnica caixa branca (estrutural), a maioria dos critérios dessa técnica é uma representação visual do código a ser testado, chamado de Grafo de Fluxo de Controle ou apenas Grafo do Programa.

Teste caixa branca, o teste estrutural



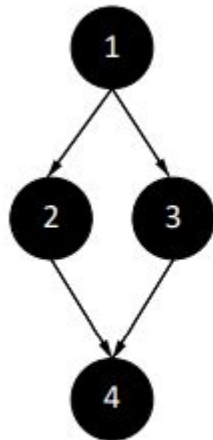


Teste caixa branca, o teste estrutural

O Grafo de Fluxo de Controle nada mais é que um conjunto de nós conectados por arcos com setas que mostram sua direção – os nós representam um bloco de comando (isso é, uma sequência atômica de linhas de código – se uma for executada, todas as outras serão) e os arcos indicam a precedência ou a transferência de controle (a sequência de execução do código fonte e seus desvios)

Teste caixa branca, o teste estrutural

Grafo de Fluxo de Controle



Detalhamento da representação de cada nó

1	1	inteiro calculo hextraPercentual(real horasTrabalhadas) {
	2	inteiro horasMes, horasExtras;
	3	horasMes = 40;
	4	SE (horasExtras > 40){

2	5	horasExtras = horasTrabalhadas - 40;
	6	return (horasExtras *100)/horasMes;

3	7	SENÃO
	8	return 0;

4	9	}
	10	}



Teste caixa branca, o teste estrutural

Sabendo como representar um programa que será testado por meio da técnica caixa branca, estamos prontos para aprender os critérios de teste dessa técnica.



Teste caixa branca, o teste estrutural

Os critérios de teste da técnica caixa branca são divididos em :

- Teste Baseado em Fluxo de Controle
- Teste Baseado em Fluxo de Dados



Teste caixa branca, o teste estrutural

Os critérios de teste da técnica caixa branca são divididos em :

- Teste Baseado em Fluxo de Controle
- Teste Baseado em Fluxo de Dados



Teste caixa branca, o teste estrutural

Os principais critérios de teste caixa branca Baseado em Fluxo de Controle são:

- Teste de Comandos
- Teste de Ramos.



Teste caixa branca, o teste estrutural

Os principais critérios de teste caixa branca Baseado em Fluxo de Controle são:

- Teste de Comandos
- Teste de Ramos.



Teste caixa branca, o teste estrutural

Teste de Comandos ou Teste Todos os nós:

Esse critério de teste define como requisito de teste que todos os comandos dos programas sejam executados ao menos uma vez. Ou seja, se olharmos para o grafo de fluxo e controle de um programa, temos que criar casos de teste de forma que todos os nós sejam exercitados ao menos uma vez.



Teste caixa branca, o teste estrutural

Teste de Comandos ou Teste Todos os nós:

Outro ponto importante para esse critério é que ao definir os casos de teste o foco deve ser nos comandos que controlam as decisões do sistema, a fim de direcionar os testes para que todos os nós sejam atingidos (Maldonado & Fabbri, 2001).



Teste caixa branca, o teste estrutural

Os principais critérios de teste caixa branca Baseado em Fluxo de Controle são:

- Teste de Comandos
- Teste de Ramos.



Teste caixa branca, o teste estrutural

Teste de Ramos ou Todas as Arestas ou Todos os Arcos:

Esse critério de teste define como requisito de teste que todas as condições verdadeiras e falsas do programa sejam executadas. Em outras palavras, todos os arcos (também chamados de arestas) do grafo de fluxo de controle devem ser exercitados, mesmo que para isso, o mesmo nó seja executado mais de uma vez (Maldonado & Fabbri, 2001).



Teste caixa branca, o teste estrutural

Os principais critérios de teste caixa branca Baseado em Fluxo de Controle são:

- Teste de Comandos
- Teste de Ramos.



Teste caixa branca, o teste estrutural

Teste de ramos:

Esse critério de teste define como requisito de teste que todos os usos das variáveis utilizadas em cada comando sejam executados. Em outras palavras, todos os arcos (também chamados de arestas) do grafo de fluxo de controle devem ser exercitados, mesmo que para isso, o mesmo nó seja executado mais de uma vez



Teste caixa branca, o teste estrutural

Todos os usos:

Para isso, devemos identificar os usos de todas as variáveis do programa (método, algoritmo), classificar o uso dessas ocorrências de acordo com *def*, *c-use* e *p-use* e então construir, para cada variável, uma tabela especificando definição e uso (par d-u), para que esses dados sejam usados na elaboração dos casos de teste .



Teste caixa branca, o teste estrutural

<i>DEF</i>	Indica definição, que é quando um valor está sendo atribuído a uma variável.
<i>C-USE</i>	Indica uso computacional, que é quando a variável é usada na avaliação de uma expressão ou comando de saída.
<i>P-USE</i>	Indica uso predicativo, que é quando a variável é usada em um predicado e consequentemente, afeta o fluxo de controle do programa.



Depuração

De forma geral podemos dizer que depurar é o processo de executar o *software* e/ou percorrer o código fonte até entender o defeito previamente encontrado e identificar o que causa o defeito para que a correção seja feita. Sendo assim, temos uma constatação importante: a atividade de depuração é consequência de um teste bem sucedido, afinal, se a depuração é necessária, é porque um defeito foi encontrado durante os testes do sistema.



Depuração

Pressman (Pressman, Engenharia de Software, 2006) menciona que a depuração tem como objetivo encontrar e corrigir a causa de um erro de software, sendo que o objetivo é alcançado por uma combinação de avaliação sistemática, intuição e sorte.



Depuração

Meyer (1979) propôs três abordagens para executar a depuração: força bruta, rastreamento e eliminação de causa:

- Força bruta;
- Rastreamento (*Backtracking*);
- Eliminação de causa.



Depuração

Meyer (1979) propôs três abordagens para executar a depuração: força bruta, rastreamento e eliminação de causa:

- Força bruta;
- Rastreamento (*Backtracking*);
- Eliminação de causa.



Depuração

Força bruta:

Nessa abordagem, o erro não é analisado sistematicamente para se descobrir a causa. Ao invés disso, tenta-se utilizar o próprio computador para a descoberta, inserindo, por exemplo, vários comandos de escrita no programa (mensagens na tela com valores internamente utilizados), listagem de memória, rastreadores de execução etc.



Depuração

Meyer (1979) propôs três abordagens para executar a depuração: força bruta, rastreamento e eliminação de causa:

- Força bruta;
- Rastreamento (*Backtracking*);
- Eliminação de causa.



Depuração

Rastreamento (*Backtracking*):

Nessa abordagem a busca pelo problema inicia-se no local em que o sintoma foi detectado e rastreia-se o software (comumente o código fonte) para trás, manualmente, até que o local da causa seja encontrado



Depuração

Meyer (1979) propôs três abordagens para executar a depuração: força bruta, rastreamento e eliminação de causa:

- Força bruta;
- Rastreamento (*Backtracking*);
- Eliminação de causa.

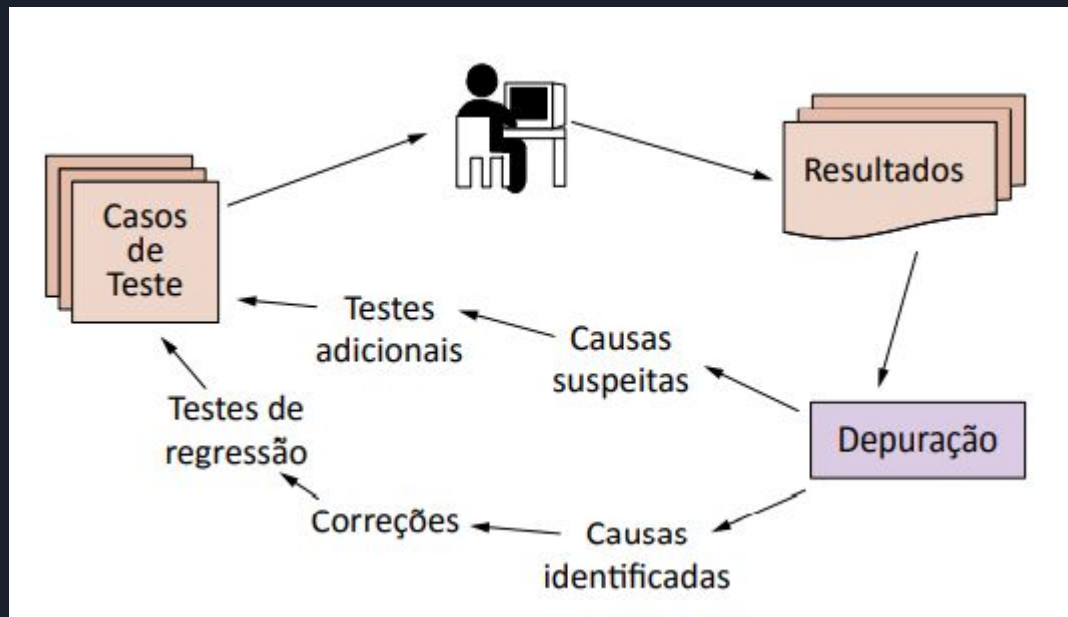


Depuração

Eliminação de causa:

Nessa abordagem a pessoa (ou o time) que está depurando o sistema supõe uma causa e casos de teste são elaborados para comprovar ou refutar essa hipótese/suposição. A suposição da causa geralmente segue o conceito de particionamento binário – os dados do defeito são particionados a fim de isolar causas em potencial e a depuração é feita a fim de confirmar ou não cada causa em potencial (hipótese).

Depuração





Depuração

Nesse processo entendemos que os testes do sistema são executados (casos de teste) e os resultados são obtidos – no caso, os resultados são os defeitos encontrados. Para cada defeito reportado o sistema deve ser depurado.



Depuração

Quando a causa (ou as causas) é identificada, o *software* é corrigido – ao fim de todas as correções, uma nova versão do sistema deve ser gerada e testes de regressão são executados a fim de identificar se ao eliminar um defeito, outros defeitos foram gerados.



Depuração

Quando a causa (ou as causas) do defeito não é identificada, causas suspeitas são levantadas, novos casos de teste são gerados e executados (abordagem – Eliminação de causa). Nesse ponto, voltamos ao início do processo, pois os novos casos de teste devem gerar novos defeitos que, nesse caso, podem ser derivações do defeito reportado inicialmente.



Teste de ambiente web

Pressman (Pressman, Engenharia de Software, 2006) salienta estratégias de teste que devem ser consideradas no ambiente web; teste de conteúdo, teste de interface, teste de navegação, teste de componente, teste de configuração, de segurança e de desempenho.



Teste de ambiente web

Teste de conteúdo:

Tem como objetivo encontrar defeitos no conteúdo, como se fosse uma revisão de um documento escrito. De fato, sistemas que envolvem muito conteúdo, em muitos casos contratam não apenas especialistas em tecnologia, mas especialistas em língua portuguesa (ou outro idioma), mas encontrar erros gramaticais, de consistência de conteúdo, de imagens etc.



Teste de ambiente web

Teste de Interface:

Tem como objetivo navegar entre os componentes de interface e validar a estética do sistema com o usuário, sempre que possível. Dessa forma, é possível identificar problemas de interação assim como inconsistências ou ambiguidades na interface.



Teste de ambiente web

Teste de Navegação:

Tem como objetivo identificar defeitos, problemas que possam existir na navegação do usuário dentre as opções do sistema. Uma possibilidade é usar o diagrama de casos de uso para exercitar cada cenário previsto na fase de projeto como parte da especificação do software.



Teste de ambiente web

Teste de Componente:

Tem como objetivo identificar defeitos, problemas que possam existir na navegação do usuário dentre as opções do sistema. Uma possibilidade é usar o diagrama de casos de uso para exercitar cada cenário previsto na fase de projeto como parte da especificação do software.



Teste de ambiente web

Teste de Configuração:

Tem como objetivo, como o próprio nome diz, identificar defeitos em relação a segurança dos componentes do sistema. Os testes tentam identificar pontos vulneráveis para que essas quebras de segurança, quando existirem, sejam corrigidas e o sistema se torne mais seguro.



Teste de ambiente web

Teste de Segurança:

Tem como objetivo é identificar os defeitos do software em um ambiente particular, geralmente o ambiente do cliente, como por exemplo, num servidor de aplicação específico. Comumente, cria-se uma matriz para mapear os possíveis sistemas operacionais, navegadores, plataformas de hardware e protocolos de comunicação e então o sistema é submetidos a testes considerando todas as variáveis.



Teste de ambiente web

Teste de Desempenho:

Tem como objetivo identificar defeitos em diversos aspectos relacionados ao desempenho do sistema, como tempo de respostas das requisições do usuário e a corretude das respostas, quando o sistema é utilizado por um alto número de usuários; como a análise do componente que prejudica o desempenho do sistema e aspecto como o impacto que problemas de desempenho tem no sistema como um todo (afinal, mesmo que um ou alguns componentes apresentem tempos de respostas elevados, isso não obrigatoriamente prejudica o desempenho do sistema como um todo).