Universidade Federal de Ouro Preto (UFOP)

CSI 403 - Engenharia de Software

ENGENHARIA DE CONFIABILIDADE

Discentes: Carlos Vitor, Luccas Vinicius, João Guilherme, Mateus Henrique, Emanuelle Ferraz, Jonas Mota

- Introdução
- Disponibilidade e Confiabilidade
- Requisitos de Confiabilidade
- Arquiteturas Tolerantes a Defeitos
- Programação para Confiabilidade
- Medição de Confiabilidade

Introdução

- Porque confiança e confiabilidade é mais importante que características funcionais..
- Os 4 Princípios de Confiança Disponibilidade, Confiabilidade, Segurança e Proteção.
- Entender a Terminologia especializada quando se discute proteção e confiança.
- Entender o que e quais erros evitar para desenvolver um sistema seguro e confiável.

Contrato

- Estagio de proposta, onde se disputa um contrato para desenvolver ou fornecer um sistema de software.

Início do Projeto

- Planejamento dos recursos que serão alocados naquele projeto. Aqui existem mais informações do que no estágio da proposta e possibilita um refinamento das estimativas de esforço.

Novos Requisitos

- Conforme os requisitos do software mudarem, a divisão de trabalho tem de ser alterada, e o cronograma precisa ser estendido.

- Planejamento inevitavelmente especulativo, pois não existe um conjunto completo de requisitos do software a ser desenvolvido.

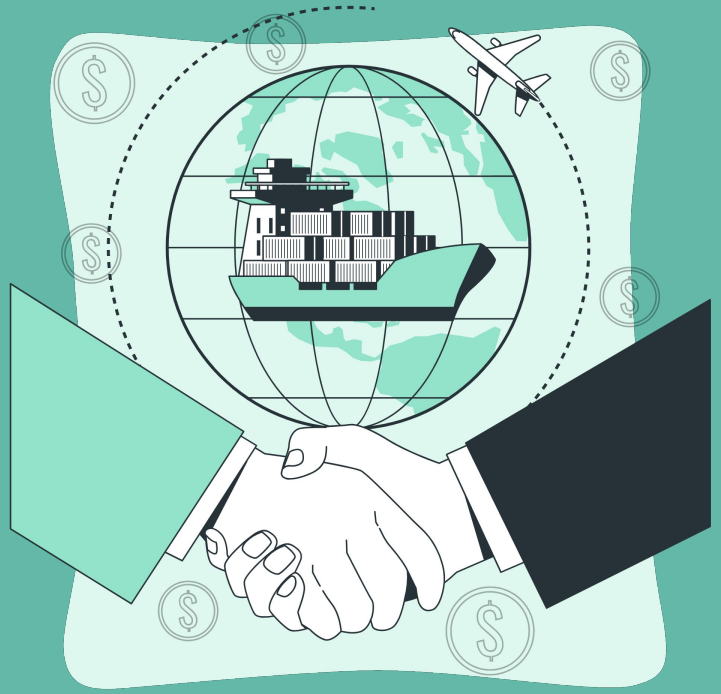
- Deve se produzir um plano viável para a realização do trabalho e um bom plano de projeto é uma parte necessária de uma proposta bem sucedida.

- Ao disputar um contrato é preciso definir o preço que será proposto ao cliente, que pode ser resumido como uma estimativa dos custos de esforço necessário para cada atividade.

- Ao ganhar o contrato, deve se então planejar o projeto novamente, levando em consideração as mudanças e as novas informações sobre o sistema.

01

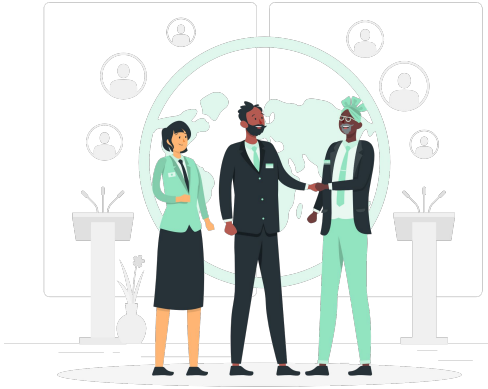
Disponibilidade e Confiabilidade



Conceitos Básicos

- **Confiabilidade:** a probabilidade de operação isenta de falhas ao longo de um período especificado, em um determinado ambiente, para um propósito específico.

- **Disponibilidade:** a probabilidade de que um sistema, em algum momento, estará operacional e será capaz de prestar serviços requisitados.



Confiabilidade

- A definição de **Confiabilidade** vai se basear em uma ideia de operação isenta de **falhas**.

- Mas afinal o que é a **falha**?

Julgamento do usuário ao utilizar o software.

Confiabilidade: Usuário -> Utilização + Ambiente

Percepções mudam de acordo com ambiente

FIGURA 11.2 Sistema como um mapeamento de entradas/saídas.

Disponibilidade

- A **Disponibilidade** não depende apenas do seu número de falhas, mas também do tempo necessário para consertar os defeitos que geraram as falhas.

- Disponibilidade e Confiabilidade estão intimamente relacionadas.

02

Requisitos de Confiabilidade

Tipos de requisitos de dependabilidade

- **Requisitos Funcionas**, que definem os recursos de verificação e recuperação que devem ser incluídos no sistema e as características que promovem a proteção contra falhas e ataques externos ao sistema.

- **Requisitos não funcionais**, que definem a confiabilidade e a disponibilidade necessárias do sistema.

Métricas de confiabilidade:

A confiabilidade pode ser especificada como uma probabilidade de ocorrência de falha do sistema quando ele estiver em uso em um ambiente operacional específico. Caso haja disposição para aceitar, por exemplo, que 1 em cada 1.000 transações possa falhar, então pode-se especificar a probabilidade de falha como 0,001. Isso não significa que haverá exatamente uma falha a cada 1.000 transações, mas, sim, que se você observar N mil transações, o número de falhas deve ser aproximadamente N.

Métricas de confiabilidade:

Três métricas podem ser empregadas para especificar a confiabilidade e a disponibilidade:

1. Probabilidade de falha sob demanda (**POFOD**, do inglês *probability of failure on demand*).
2. Taxa de ocorrência de falhas (**ROCOF**, do inglês *rate of occurence of failures*).

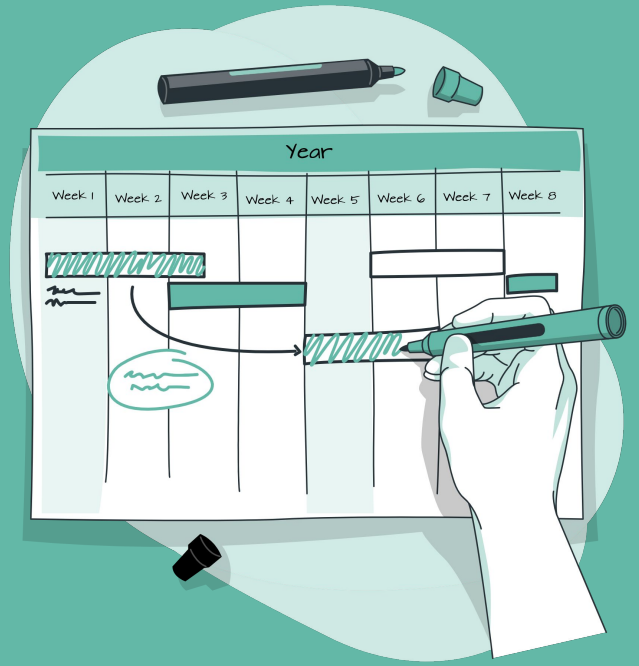
3. Disponibilidade (**AVAIL**, do inglês *availability*).

Requisitos de confiabilidade não funcionais:

- Utilizam das métricas apresentadas anteriormente para definir uma meta para o sistema.
- Ajudam a esclarecer as necessidades reais dos stakeholders.
- São usados como base para a realização de testes, sendo possivel parar os testes e concluir assim que o sistema alcançar o nível de confiabilidade necessário.
- É um modelo de avaliação para diferentes estratégias de projeto destinadas a melhorar a confiabilidade de um sistema.
- Importante nas verificações de desempenho de um sistema, para evidenciar se a meta de confiabilidade exigida foi cumprida.

03

Arquiteturas tolerantes a defeitos



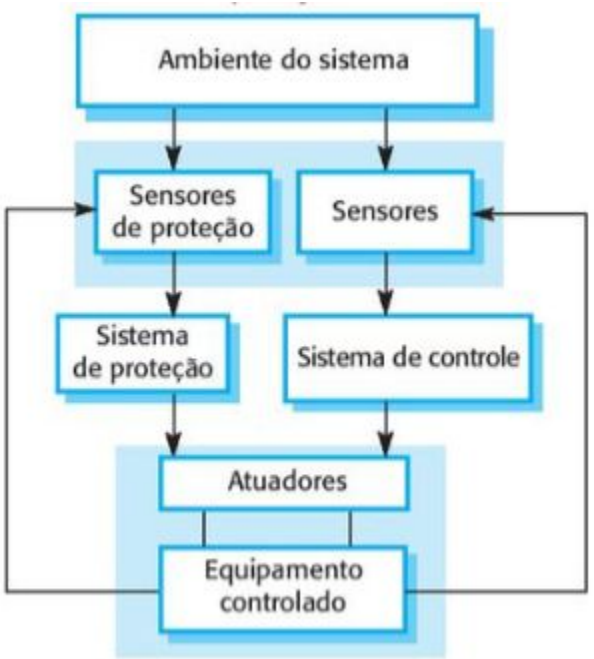
Definição

- ➔ Abordagem usada para que os sistemas consigam continuar suas operações mesmo que algum defeito de hardware ou software, e estado de sistema errado.
- ➔ Corrigir os defeitos antes que se tornem falhas.
- ➔ Necessário em sistemas críticos em segurança, segurança da informação e softwares que não passam para estados seguros sem correção de erro.
- ➔ Sistemas redundantes e diversos.

Sistemas de proteção

- ➔ Sistema especializado que atua junto a outro sistema.
- ➔ Monitoram ações de outros sistemas.
- ➔ Caso alguma ação esperada não seja tomada, acionam medidas de proteção.
- ➔ Monitoram independentemente o seu ambiente.
- ➔ Contém apenas funcionalidade críticas necessárias.
- ➔ A vantagem é que o sistema de proteção pode ser mais simples que o monitorado.

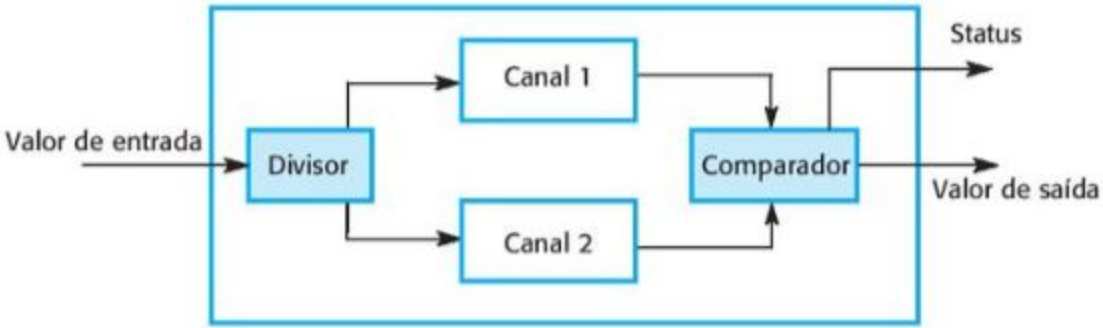
Sistemas de proteção



Automonitoramento

- O próprio sistema monitora sua atividade .
- Caso um problema seja detectado, o sistema toma alguma medida para corrigir o defeito.
- Os cálculos do sistema são feitos separadamente.
- Caso o resultado não coincida, o sistema toma uma medida de segurança.

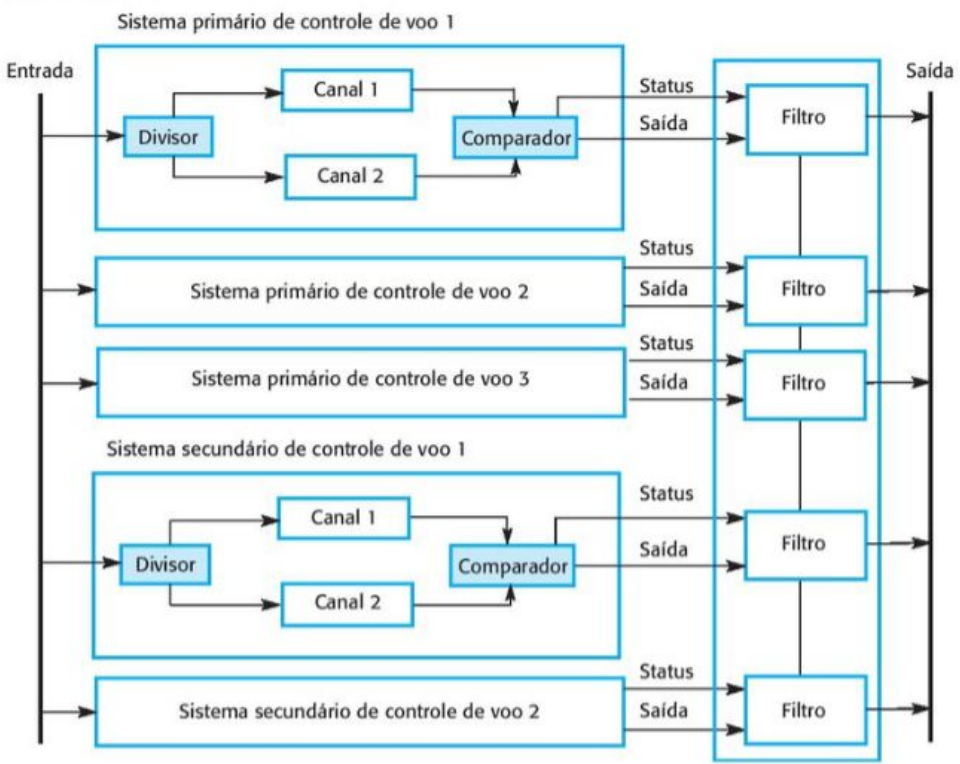
Automonitoramento



Automonitoramento

- Para ter uma boa eficaz, é necessário que hardwares e softwares diversificados sejam usados.
- Geralmente, esse tipo de abordagem é usada em sistemas que a precisão é mais importante que a disponibilidade do sistema.
- Em sistemas que a disponibilidade é importante, o automonitoramento pode ser feito com sistemas paralelos.
- Nesse caso, uma decisão de qual dos cálculos deve ser usado tem de ser tomada.

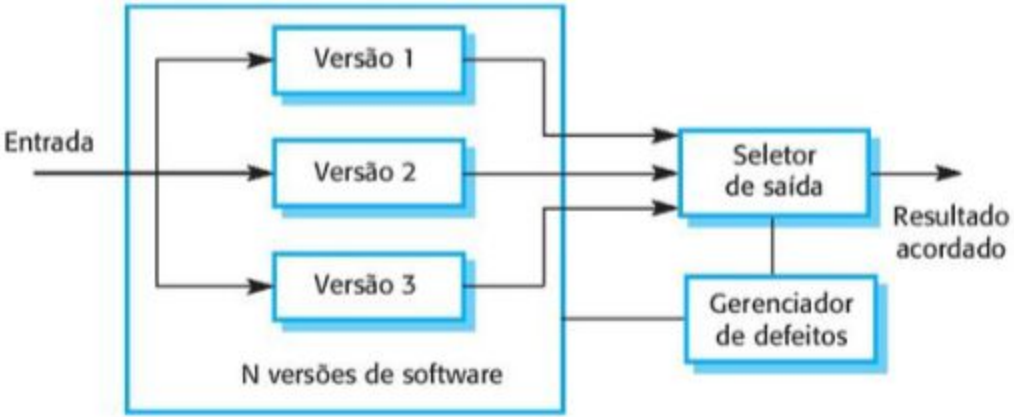
Automonitoramento



Programação em N-versões

- Vem da ideia de desenvolvimento de hardware TMR (redundância modular tripla).
- Essa abordagem já é usada no desenvolvimento de hardware e foi passada para o de software.
- O software é replicada 3 vezes.
- Cada saída é comparada e uma é escolhida.
- Se duas ou mais saídas forem iguais, esse é o retorno.
- Caso um dos softwares tenha defeito, um gerenciador pode concerta-lo.

Programação em N-versões



Diversidade de software

➔ Todos os métodos anteriores dependem da diversidade de software.

➔ Esse método é adotado para que cada parte do software seja independente e falhe de maneira diferente.

➔ A empresa pode incluir requisitos específicos como:

- Diferentes métodos sejam usados.
- Cada parte do sistema use uma linguagem diferente.
- Cada parte do sistema seja desenvolvido em um ambiente diferente.
- Utilização de diferentes algoritmos com mesmo papel em cada parte do software.

Diversidade de software

➔ Na prática, é impossível ter uma independência completa dos softwares.

➔ Isso se deve por alguns motivos, como:

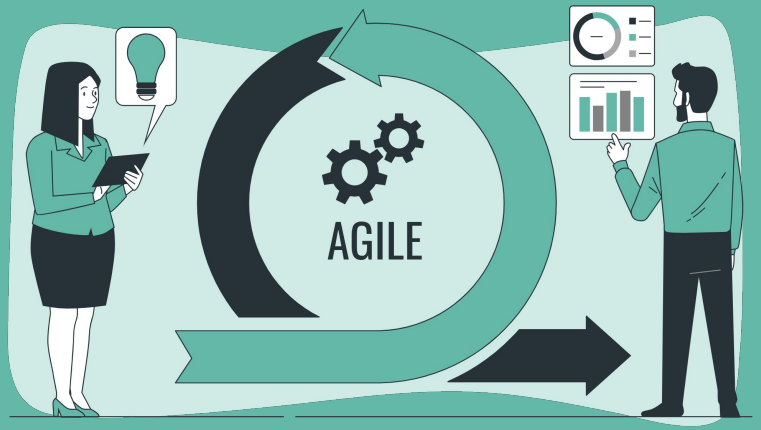
- Membros de diferentes equipes podem ter a mesma cultura ou terem recebido a mesma abordagem de aprendizado.
- Requisitos incorretos ou baseados em interpretação pessoal.

➔ Uma maneira de inibir isso é desenvolver especificações detalhadas, com menos ambiguidade.

➔ Além de mudar a forma que cada equipe trabalha.

04

Planejamento Ágil



O que são métodos ágeis?

- Os **métodos ágeis de desenvolvimento de software** são abordagens interativas nas quais o software é **desenvolvido e entregue aos clientes em incrementos.** A funcionalidade desses incrementos não é planejada com antecedência, mas decidida durante o desenvolvimento.

“Prioridades e as necessidades do cliente mudam e, assim, faz sentido ter um plano flexível capaz de acomodar essas mudanças”

O Manifesto ágil

- O Manifesto Ágil é uma declaração de valores e princípios essenciais para o desenvolvimento de software

Extreme Programming (XP)

Extreme Programming (XP)

05

Técnicas de Estimativa

- As estimativas de cronograma de projeto são **difíceis.**

- Existem tantas incertezas que é **impossível** estimar com precisão os custos de desenvolvimento de sistema durante os estágios iniciais de um projeto.
  Existe até uma dificuldade fundamental na avaliação da precisão das diferentes abordagens para a estimativa de custo e esforço.

- A estimativa é usada para definir o orçamento de projeto e o produto é ajustado para que o orçamento seja realizado. Um projeto que está dentro do orçamento pode ter alcançado isso em detrimento dos recursos do software que está sendo desenvolvido

1. Técnicas baseadas em experiências: As estimativas de requisitos de futuros esforços se baseiam na experiência do gerente em projetos anteriores e em seu domínio de aplicação. Essencialmente, o gerente faz uma avaliação informada do que os requisitos de esforço podem ser.

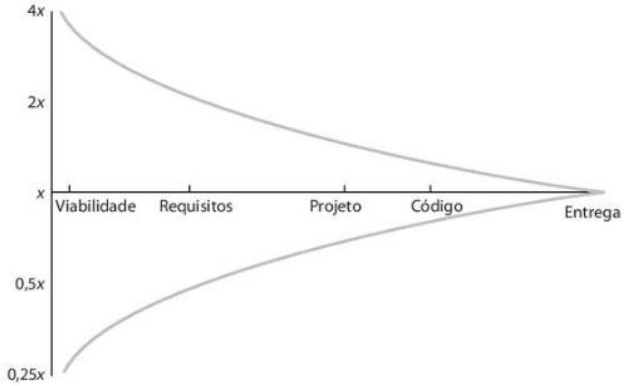
2. Modelagem algorítmica de custos: Uma abordagem usada em muitas situações para calcular o esforço de projeto com base em estimativas de atributos de produto, como tamanho e características do processo, por exemplo a experiência do pessoal envolvido.

$$Esforco = A \cdot Tamanho^{B} \cdot M$$

- ***A*** é um fator constante que depende das práticas organizacionais locais e do tipo de software em desenvolvimento.
- ***Tamanho*** pode ser uma avaliação do tamanho do código do software.
- ***B*** encontra-se geralmente entre 1 e 15.
- ***M*** é um multiplicador feito pela combinação de atributos de processo, produtos e de desenvolvimento, como os requisitos de confiança para o software e a experiência da equipe de desenvolvimento.

Em ambos os casos, você precisará usar seu julgamento para estimar o esforço, ou estimar as características de projeto e de produto. Na fase de iniciação de um projeto, essas estimativas têm uma grande margem de erro.

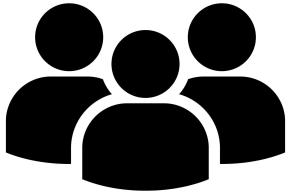
Figura 23.6 Incerteza de estimativa



Pontos importantes

O preço cobrado por um sistema não depende apenas dos custos estimados de desenvolvimento e do lucro exigido pela empresa de desenvolvimento. Fatores organizacionais podem significar que o preço seja aumentado para compensar riscos altos, ou reduzido para ganhar vantagem competitiva.
Muitas vezes, o preço do software é definido para ganhar um contrato e, em seguida, a funcionalidade do sistema é ajustada para atender ao preço estimado.

Geralmente, é útil contar com um grupo de pessoas envolvidas na estimativa de esforço e pedir a cada membro do grupo para explicar sua estimativa. Isso muitas vezes revela fatores que outros não levaram em consideração. Em seguida, você itera a uma estimativa acordada pelo grupo.



05

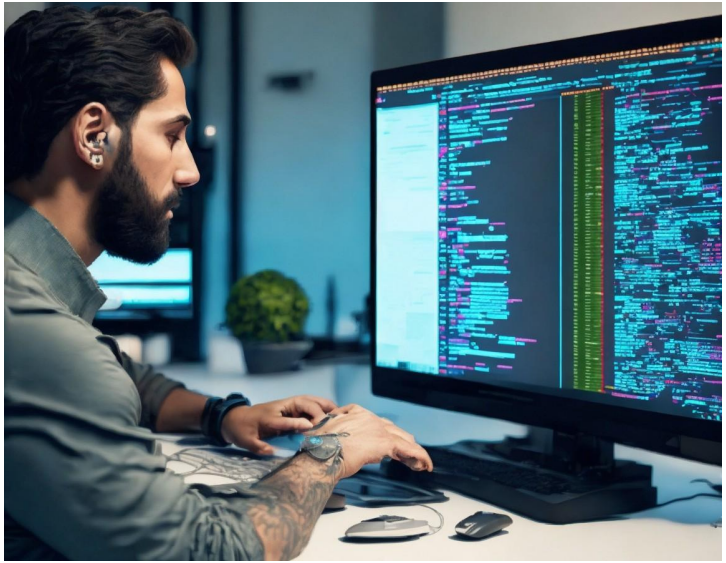
Programação para confiabilidade

Programação para Confiabilidade

Em um cenário onde a estabilidade e a segurança dos sistemas são cruciais, a engenharia de confiabilidade se destaca como um pilar fundamental no desenvolvimento de software. Neste contexto, destacamos oito práticas recomendadas que transcendem barreiras de linguagem de programação, visando reduzir erros nos sistemas entregues e fortalecer a segurança da informação. A seguir, exploraremos essas diretrizes que, quando seguidas, proporcionam robustez e confiabilidade aos nossos projetos.

8 Práticas Recomendadas

- Limitar a visibilidade das informações no programa:
- Conferir a validade de todas as entradas:
- Fornecer tratamento para todas as exceções:
- Minimizar o uso de construtos propensos a erro:
- Fornecer capacidade de reinicialização:
- Conferir limites do vetor:
- Incluir tempos de espera (timeouts) ao invocar componentes externos:
- Dar nomes a todas as constantes que representam valores do mundo real:



06

Medição da Confiabilidade

Para avaliar a confiabilidade de um sistema, é preciso coletar dados sobre sua operação. Os dados necessários podem incluir:

1- Número de falhas do sistema, dado um número de solicitações ou eventos de um sistema. Isso serve para medir a POFOD (Probabilidade de Falha Sob Demanda) e se aplica independentemente do tempo em que as demandas são feitas.

2- O tempo ou número de transações entre falhas do sistema somado ao tempo decorrido total ou ao número total de transações. Isso serve para medir a Taxa de Ocorrência de Falhas) e MTTF (Tempo Médio até a Falha)

3- O tempo de reparo ou reinicialização após uma falha do sistema que leva à perda do serviço. Serve para medir a disponibilidade, que não depende apenas do tempo entre as falhas, mas também do tempo necessário para o sistema voltar a operar.

As unidades de tempo que podem ser utilizadas nessas métricas são o tempo do calendário ou uma unidade discreta, como o número de transações.

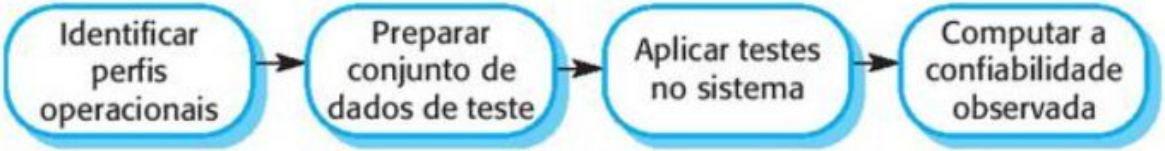
O tempo do calendário deve ser usado para os sistemas que estão em operação contínua como Sistemas de monitoramento ou os de controle de processos

Portanto o ROCOF, poderia ser o número de falhas por dia.

Os sistemas que processam transações, como os caixas eletrônicos de bancos ou sistemas de reserva de passagem aéreas, têm acessos variáveis dependendo do horário do dia. Nesses casos, a unidade de ‘tempo’ utilizada poderia ser o número de transações ou o número de transações defeituosas por N-mil transações.

O teste de confiabilidade é um processo de teste estático que visa medir a confiabilidade de um sistema, as métricas de confiabilidade, como a POFOD e a ROCOF, podem ser utilizadas para especificar quantitativamente e a confiabilidade do software.

FIGURA 11.13 Teste estatístico para medição da confiabilidade.



Principais dificuldades:

- **Incerteza do perfil operacional.** Os perfis operacionais baseados na experiência com outros sistemas podem não ser um reflexo exato do uso real do sistema
- **Possibilidade de altos custos para a geração de dados de teste.**
- **Incerteza estática quando uma alta confiabilidade é especificada.** Deve-se gerar um número de falhas estaticamente significativo para permitir as medições exatas da confiabilidade. Quando o software já é confiável, ocorrem relativamente, poucas falhas e é difícil gerar novas falhas.
- **Reconhecimento da falha**. Nem Sempre é óbvio se uma falha do sistema ocorreu ou não.

Referências

SOMMERVILLE, Ian. Engenharia de software. 10. ed. São Paulo: Pearson, 2018. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 21 nov. 2023.

OBRIGADO!