Fatec
Guaratinguetá

Tecnologia em Análise e Desenvolvimento de Sistemas

Engenharia de Software III

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Prof. Claudemir Santos Pinto
profdemir@yahoo.com.br

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Conceitos**

Sistemas críticos são sistemas onde as falhas podem resultar em perdas econômicas significativas, danos físicos ou ameaças à vida humana.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- São divididos, basicamente, em três tipos: sistemas críticos de segurança, de missão e de negócios.

A principal diferença entre eles é o tipo de prejuízo que pode ser causado por falhas. No sistema de segurança uma falha pode causar riscos a vida humana ou danos ao meio ambiente, no sistema de missão os riscos são quanto a problemas no alcance de uma meta, no sistema de negócios uma falha pode causa grandes prejuízos financeiros.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Abordagens para desenvolver um sistema confiável:

- Prevenção de defeitos
- Detecção de defeitos
- Tolerância a defeitos

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Prevenção de defeitos:

- O processos de projeto e implementação do sistema devem usar abordagens de desenvolvimento de software que ajudem a evitar erros de programação, e assim, minimizar o número de defeitos de um programa.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Detecção de defeitos:

- Os processos de verificação e de validação são projetados para descobrir e remover defeitos de um programa antes que este seja implantado para uso operacional

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Tolerância a defeitos

- O sistema é projetado de forma que os defeitos ou o comportamento inesperado do sistema, durante a execução, sejam detectados e gerenciados de modo que a falha do sistema não ocorra.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- Sistemas críticos podem incluir componentes que replicam a funcionalidade de outros componentes (redundância) ou código adicional de verificação que não é estritamente necessário para que o sistema funcione. Portanto, os defeitos podem ser detectados antes que causem falhas, e o sistema pode ser capaz de continuar operando caso os componentes individuais falhem.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Para sistemas de pequeno e médio portes, as técnicas de engenharia de software provavelmente tornam possível desenvolver software livre de defeitos. Para atingir esse objetivo você precisa usar uma gama de técnicas de engenharia de software:

Processos de software confiáveis

Gerenciamento de qualidade

Especificação formal

Verificação estática

Programação segura

Informações protegidas

O uso de um processo de software confiável com atividades de validação e verificação apropriadas é essencial caso o número de defeitos em um programa deva ser minimizado e aqueles ignorados devam ser detectados

- Processos de software confiáveis
- Gerenciamento de qualidade
- Especificação formal
- Verificação estática
- Programação segura
- Informações protegidas

Técnicas de verificação estática, como o uso de analisadores estáticos, podem encontrar características anômalas de programação que podem ser defeitos.

A organização que desenvolve o sistema deve ter uma cultura na qual a qualidade conduz o processo de software. A cultura deve estimular os programadores a criar programas livres de defeitos. Padrões de desenvolvimento e projeto devem ser estabelecidos

Deve haver uma especificação precisa de sistema que defina o sistema a ser implementado. Muitos defeitos de projeto e programação resultam de uma interpretação errada de uma especificação ambígua ou mal escrita

Algumas construções da linguagem de programação são mais complexas e propensas a erros caso sejam usadas. Programação segura significa evitar o uso ou tomar maior cuidado no uso dessas construções

Deve ser usada uma abordagem para projeto e implementação de software baseada em ocultamento e encapsulamento de informações.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Programação confiável**

Envolve o uso de construções e técnicas de programação que contribuem para prevenção e tolerância a defeitos.

Essas técnicas baseiam-se no fato de que há uma distinção entre defeitos e falhas: uma falha é algo observável pelos usuários de um sistema, enquanto um defeito é uma característica interna do sistema. Se um defeito se manifesta na execução de um programa, você deve ser capaz de tolerá-lo, detectando-o e executando uma ação de recuperação antes que isso resulte em uma falha do sistema.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Tolerância a defeitos**

Um sistema tolerante a defeitos pode continuar em operação após terem ocorrido alguns defeitos do sistema. Os mecanismos de tolerância a defeitos de um sistema asseguram que esses defeitos não causem falha do sistema. A tolerância a defeitos pode ser necessária em situações em que a falha do sistema poderia causar um acidente catastrófico ou em que uma perda da operação do sistema causaria grandes prejuízos econômicos. Por exemplo, os computadores de uma aeronave devem continuar trabalhando até ela ter pousado; os do controle de tráfego aéreo também.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Existem 4 aspectos da tolerância a defeitos:

- Detecção de defeitos
- Avaliação de danos
- Recuperação de defeitos
- Reparação de defeitos

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- Detecção de defeitos

O sistema deve detectar um defeito que pode conduzir a uma falha de sistema. Em geral, isso envolve a verificação se o sistema é consistente. No caso de haver uma inconsistência, uma variável de estado tem seu valor alterado.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- Avaliação de danos

As partes do sistema afetadas pelo defeito devem ser detectadas. O dano pode somente ser avaliado se for possível aplicar alguma "função de validação" que verifique se o estado é consistente. Se forem encontradas inconsistências, elas serão destacadas ou sinalizadas de alguma maneira.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- Recuperação de defeitos

O sistema deve restaurar seu estado para o estado "seguro" conhecido. Esse estado pode ser atingido pela correção do estado de dano ou pela restauração do sistema para um estado "seguro" conhecido.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- Reparação de defeitos

Envolve a modificação do sistema de maneira que a falha não ocorra. Entretanto, muitos defeitos de software manifestam-se como estados transitórios. Eles são devidos a uma combinação peculiar de entradas do sistema. Nenhum reparo é necessário e o processamento normal pode ser retomado imediatamente após a recuperação do defeito.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Pontos chave:

- A confiança em um programa pode ser obtida evitando-se a introdução de defeitos. Isso se consegue pela detecção e remoção de defeitos antes da implantação do sistema e pela inclusão de recursos de tolerância a defeitos que permitem ao sistema permanecer em operação mesmo depois que um defeito tenha ocorrido.
- O uso de redundância e de diversidade nos processos e nos sistemas de software é essencial para o desenvolvimento de sistemas confiáveis.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Pontos chave:

- O uso de um processo bem definido é importante para os defeitos de um sistema serem minimizados. O processo deve incluir atividades de verificação e validação de todos os estágios, desde a definição de requisitos até a implementação do sistema.
- Você **JAMAIS** deve utilizar-se de "gambiarras" em programação, especialmente se for em sistemas críticos.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

Pontos chave:

- As exceções são usadas para apoiar o gerenciamento de erros em sistemas confiáveis. Todas as exceções devem ser explicitamente tratadas em um sistema crítico.
- Os quatro aspectos de tolerância a defeitos de programa são: detecção de falhas, avaliação de danos, recuperação e reparação de defeitos.
- Programação em N-versões e blocos de recuperação são abordagens alternativas para arquiteturas tolerantes a defeitos, nas quais cópias redundantes do hardware e software são mantidas.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Confiança no sistema**

Para sistemas críticos a propriedade de sistema mais importante é a **confiança**.

A confiança de um sistema reflete o grau de confiança do usuário nesse sistema, bem como a extensão de confiança do usuário que o operará conforme suas expectativas e que não 'falhará' durante o uso normal.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Confiança no sistema**

A confiança em um sistema equivale ao seu merecimento de confiança.

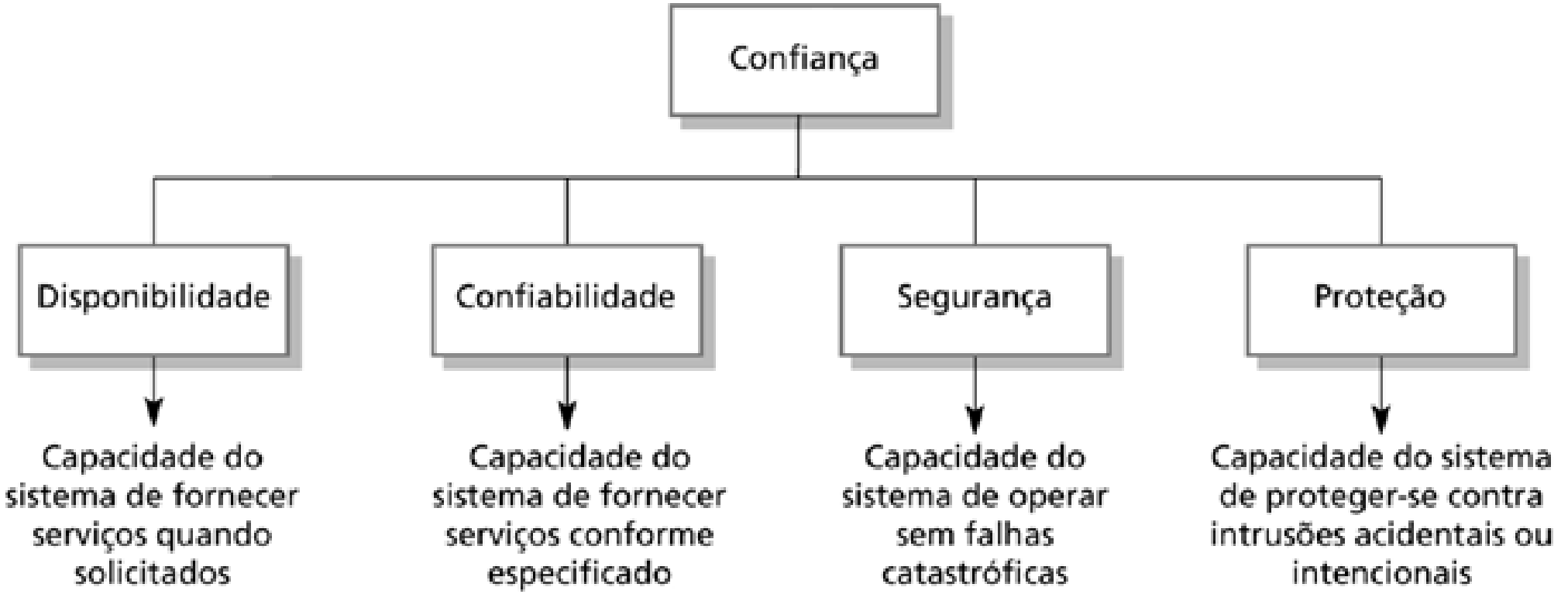
Um sistema confiável é aquele em que os usuários depositam sua confiança.

As principais dimensões de confiança são:

- Disponibilidade;
- Confiabilidade;
- Segurança;
- Proteção.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- Confiança no sistema



DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **A importância da confiança**

Sistemas que não são confiáveis, que são inseguros ou desprotegidos, podem ser rejeitados pelos seus usuários.

Os custos com falha de sistema podem ser muito altos.

Sistemas não confiáveis podem causar perda de informação e, consequentemente, um alto custo de recuperação.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Outras propriedades de confiança**

Facilidade de reparo

Reflete a amplitude em que o sistema pode ser reparado no caso de uma eventual falha;

Facilidade de manutenção

Reflete a amplitude em que o sistema pode ser adaptado para novos requisitos;

Capacidade de sobrevivência

Reflete a amplitude na qual o sistema pode fornecer serviços quando está sob ataque hostil ou sob falha.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Razões das falhas**

Falhas de hardware

O hardware pode falhar por causa de erros de projeto e de produção, ou pelo fato de os componentes terem atingido o fim de sua vida útil.

Falhas de software

Software falha devido a erros na sua especificação, projeto ou implementação.

Falha operacional

Seres humanos cometem erros. Atualmente, talvez sejam a maior causa de falhas de sistema.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Confiança versus desempenho**

Sistemas não confiáveis podem ser rejeitados por seus usuários, pois podem causar perda de informação valiosa.

Porém, a confiança cobra seu preço em relação ao desempenho. Geralmente o código fica maior, o tempo de acesso às informações também é maior.

É muito difícil ajustar sistemas para torná-los mais confiáveis sem alterar o desempenho.

Custos de confiança

Custos de confiança tendem a aumentar exponencialmente quando níveis cada vez mais altos de confiança são requisitados.

Existem duas razões para isso:

- O uso de técnicas de desenvolvimento mais onerosas e de hardware que são requisitados para atingir os níveis mais altos de confiança
- O aumento de testes e a validação de sistema que é necessária para convencer o cliente do sistema de que os níveis de confiança requisitados foram atingidos.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Melhoria de confiabilidade**

A remoção de X% de defeitos de um sistema não melhorará, necessariamente, a confiabilidade em X%. Um estudo da IBM mostrou que a remoção de 60% de defeitos de produtos resultaram em uma melhoria de 3% da confiabilidade.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Melhoria de confiabilidade**

Defeitos de programa podem estar em seções raramente executadas do código e, desse modo, nunca serem encontrados pelos usuários. A remoção destes defeitos não afetam a percepção da confiabilidade.

Um programa com defeitos conhecidos pelos desenvolvedores podem, portanto, ainda ser vistos como confiáveis pelos seus usuários.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Terminologia de segurança**

Termo	Descrição
Acidente (ou desgraça)	Evento ou seqüência de eventos não planejados que resulta em morte ou ferimento de humanos, danos à propriedade ou ao ambiente. Uma máquina controlada por computador que fere seu operador é um exemplo de um acidente.
Perigo	Condição com potencial para causar ou contribuir para um acidente. A falha de um sensor que detecta um obstáculo em frente de uma máquina é um exemplo de perigo.
Dano	Medida de perda resultante de um acidente. Um dano pode variar desde a morte de várias pessoas como resultado de um acidente até ferimentos de pouca importância ou danos à propriedade.
Severidade do perigo	Avaliação do pior dano possível que poderia resultar de determinado perigo. A severidade do perigo pode variar de catastrófica, na qual várias pessoas são mortas, até somente danos de pouca importância.
Probabilidade de perigos	Probabilidade de ocorrência de eventos que criam um risco. Valores de probabilidade tendem a ser arbitrários, mas variam de *provável* (digamos, chance de 1/100 de ocorrência de um risco) a *implausível* (não existem situações concebíveis nas quais o perigo possa ocorrer).
Risco	É a medida da probabilidade de que o sistema causará um acidente. O risco é avaliado considerando-se a probabilidade do perigo, a severidade do perigo e a probabilidade de que o perigo resultará em um acidente.

DESENVOLVIMENTO DE SISTEMAS CRÍTICOS

- **Terminologia de proteção**

Termo	Descrição
Exposição	Possível perda ou dano no sistema computacional. Pode ser perda ou danos nos dados ou pode ser perda de tempo ou esforço, se a recuperação é necessária após uma brecha na proteção.
Vulnerabilidade	Uma fraqueza no sistema baseado em computador que pode ser explorada para causar perda ou dano.
Ataque	Uma exploração da vulnerabilidade do sistema. Geralmente parte de fora do sistema e é uma tentativa deliberada para causar algum dano.
Ameaças	Circunstâncias que têm potencial para causar perda ou dano. Você pode pensar nelas como uma vulnerabilidade do sistema que está sujeita a um ataque.
Controle	Uma medida de proteção que reduz uma vulnerabilidade do sistema. Criptografia pode ser um exemplo de controle que reduz a vulnerabilidade de um sistema fraco de controle de acesso.

Atividade de fixação

1. Diferencie as quatro dimensões de confiança num sistema crítico: **Disponibilidade**, **Confiabilidade**, **Segurança** e **Proteção**

2. Cite 3 motivos pelos quais justifica-se o custo de as empresas assegurarem que seu software é livre de defeitos.

3. Explique porque a remoção de X% de defeitos de um sistema não melhora, necessariamente, a confiabilidade em X%.

Tarefa agendada via Teams.