

MATÉRIA - GESTÃO E QUALIDADE DE SOFTWARE – GQS

PROFESSOR - Calvetti

ALUNO - João Luiz da Silva – RA 82420546

Atividade 2:

1. Tabela Comparativa Falha X Causa X Prevenção (Exemplo Ariane 5)

Essa tabela deve ser criada para cada caso discutido no seu material. O exemplo dado com o **Ariane 5** mostra um caso onde uma falha ocorreu devido a uma falha de conversão numérica, que foi causada pela reutilização de software do Ariane 4 sem a devida adaptação. A prevenção envolveu testes de simulação e auditorias para garantir que o software atendesse às novas condições.

Para outros casos que você deve analisar, basta seguir a estrutura:

- **Falha:** Descrever a falha.
- **Causa:** Explicar a origem ou motivo da falha.
- **Prevenção:** Apresentar as ações que poderiam ter evitado essa falha, como testes adicionais, revisão de requisitos, auditorias, etc.

Exemplo de outra falha:

Falha: Knight Capital – Erro na execução de ordens de ações em massa.

Causa: Código legado não desativado corretamente, conflito entre novo código e código antigo.

Prevenção: Testes de integração, testes de regressão, auditoria de código.

2. Ligação com a ISO/IEC 25010 – Qualidade de Produto

Aqui, você deve conectar as falhas e causas observadas em cada caso com as características de qualidade do software segundo a **ISO/IEC 25010**, que define as qualidades desejadas de um produto de software.

Exemplo:

- **Caso Knight Capital**
Característica ISO/IEC 25010: Confiabilidade (Reliability)
Problema Observado: Falha no comportamento consistente do sistema durante operações de carga elevada.
Consequência: Ordens erradas em massa enviadas à bolsa de valores.

Você vai fazer o mesmo para os outros casos citados, destacando qual característica de qualidade foi afetada pela falha.

3. Ligação com CMMI/MPS.BR – Qualidade de Processo

Aqui, você vai identificar como falhas específicas estão ligadas a áreas de processo do **CMMI (Capability Maturity Model Integration)** ou **MPS.BR**, que são modelos de maturidade para gestão de processos.

Exemplo:

- **Caso Therac-25**
Área de Processo CMMI/MPS.BR: Gerência de Requisitos (REQM)
Problema Observado: Requisitos críticos de segurança não especificados detalhadamente.
Como o Processo Poderia Evitar a Falha: Especificação detalhada de requisitos funcionais e de segurança com rastreabilidade.

Você pode aplicar a mesma lógica para outros casos como **Ariane 5, Knight Capital**, etc.

4. Importância dos Testes e Auditorias

Para cada caso, você vai descrever como os testes poderiam ter evitado ou minimizado os danos causados pela falha.

Exemplo:

- **Caso Knight Capital**

Tipo de Teste: Testes de Integração

Como ajudaria: Detectariam conflitos entre o novo código e o legado, evitando que ordens erradas fossem executadas.

Este item é importante para entender como uma abordagem proativa de testes (integração, regressão, simulação, etc.) poderia ter evitado os erros.

5. Papel da Gestão de Requisitos

A gestão de requisitos é crucial para evitar falhas de software, e essa seção vai detalhar como cada uma das falhas poderia ser mitigada com uma boa gestão de requisitos.

Exemplo:

- **Caso Therac-25**

Etapa da Gestão de Requisitos: Elicitação de Requisitos

Exemplo Aplicado: Identificar todas as condições de segurança para evitar exposição excessiva de radiação.

Benefício: Garantir que todos os cenários críticos sejam considerados desde o início, evitando falhas no sistema.

Cada caso precisa de uma análise do impacto de uma boa gestão de requisitos, desde a elicitação até a documentação e gestão de mudanças.

Essas atividades exigem que você conecte falhas de sistemas reais a conceitos teóricos, normas de qualidade, e melhores práticas de gestão de software. Se você quiser, posso ajudar a elaborar as respostas específicas para cada uma dessas partes, apenas me diga qual caso você gostaria de analisar em detalhes primeiro.

Você disse:

pode ser

O ChatGPT disse:

Ótimo! Vamos começar com a primeira parte, que é a **Tabela Comparativa Falha x Causa x Prevenção**. Para isso, vamos focar em alguns dos casos mais conhecidos que você mencionou. Vou montar uma tabela comparativa para cada um, com base nas informações fornecidas.

1. Tabela Comparativa Falha x Causa x Prevenção

Caso Ariane 5 (1996)

Falha	Causa	Prevenção
Explosão do foguete Ariane 5	Erro de conversão numérica (overflow): tentativa de converter um valor de 64 bits (velocidade horizontal maior do que	- Revisão de requisitos considerando novas

Falha	Causa	Prevenção
Logo após o lançamento.	a prevista no Ariane 4) para 16 bits. O software reutilizado do Ariane 4 não foi adaptado para os novos limites.	condições de operação. - Testes de simulação com cenários extremos. - Validação de limites numéricos em variáveis críticas. - Auditoria independente de software reutilizado.

Caso Knight Capital (2012)

Falha	Causa	Prevenção
Perda de \$440 milhões devido a ordens erradas enviadas à bolsa.	Conflito entre o novo código e o legado ("Power Peg"). O código não foi desativado corretamente, resultando em ordens não intencionais sendo executadas em massa.	- Testes de integração para verificar compatibilidade entre novo e velho código. - Testes de regressão para garantir que funcionalidades antigas não fossem ativadas por engano. - Auditoria rigorosa do código legado e novas implementações.

Caso Therac-25 (1985-1987)

Falha	Causa	Prevenção
Superexposição de radiação a pacientes, causando mortes e danos.	Falha na especificação de requisitos críticos de segurança. O software não possuía intertravamentos e não era capaz de detectar condições perigosas.	- Especificação detalhada dos requisitos funcionais e de segurança, incluindo intertravamentos. - Implementação de validação rigorosa em ambiente real com testes simulados. - Adoção de práticas de verificação formal.

2. Ligação com a ISO/IEC 25010 – Qualidade de Produto

Agora, vamos ligar os casos à **ISO/IEC 25010**, que define várias características de qualidade de software. Vamos pegar os casos já discutidos:

Caso Knight Capital (2012)

Característica ISO/IEC 25010	Problema Observado no Caso	Consequência
Confiabilidade (Reliability)	Falta de comportamento consistente sob condições normais e carga elevada.	Ordens erradas em massa foram enviadas à bolsa, causando uma perda de \$440 milhões.
Manutenibilidade (Maintainability)	Código legado não foi removido ou desativado corretamente, causando falha.	Ativação indevida de funções obsoletas.
Segurança (Security)	Ausência de mecanismos de controle de execução, como rollback seguro.	Não foi possível interromper rapidamente o sistema para evitar a execução das ordens erradas.
Compatibilidade (Compatibility)	Novo código entrou em conflito com rotinas antigas não eliminadas.	Comportamento inesperado, levando a ordens erradas sendo disparadas.
Caso Therac-25 (1985-1987)		
Característica ISO/IEC 25010	Problema Observado no Caso	Consequência
Confiabilidade (Reliability)	O software falhou em garantir a segurança operacional dos pacientes.	Exposição excessiva de radiação, resultando em danos e mortes.
Manutenibilidade (Maintainability)	O sistema não foi mantido adequadamente para lidar com novos cenários de segurança.	Impossibilidade de atualização ou correção rápida durante a operação.
Segurança (Security)	Falta de intertravamentos e checagens de segurança.	Pacientes foram irradiados com doses fatais, sem a possibilidade de parada imediata.
Compatibilidade (Compatibility)	O software não era compatível com novos sistemas de segurança ou tecnologias de hardware.	Falha em verificar as condições operacionais de forma adequada.

3. Ligação com CMMI/MPS.BR – Qualidade de Processo

Agora, vamos explorar o **CMMI/MPS.BR** e como a qualidade de processo poderia ter evitado as falhas.

Caso Therac-25

Área de Processo CMMI/MPS.BR	Problema Observado no Caso	Como o Processo Poderia Evitar a Falha
Gerência de Requisitos (REQM)	Requisitos de segurança e intertravamentos não foram especificados corretamente.	Especificação de requisitos claros de segurança e intertravamentos, com rastreabilidade para garantir a implementação correta.
Verificação (VER)	Testes de software incompletos, sem simulação de todas as condições operacionais.	Implementação de processos de verificação formal, incluindo testes em condições extremas e simuladas.
Validação (VAL)	O software não foi validado em um ambiente real antes da liberação.	Realização de testes em ambiente simulado realista (ex: com pacientes simulados) para validar as funcionalidades de segurança do sistema.
Gestão de Configuração (CM)	O código legado não foi revisado adequadamente.	Uso de controle de versões, revisão de código e auditorias formais antes de qualquer integração ou atualização.
Gerência de Riscos (RSKM)	Riscos críticos não foram antecipados nem mitigados.	Identificação de riscos críticos relacionados à segurança, com desenvolvimento de planos de mitigação e contingência.

4. Importância dos Testes e Auditorias

Agora, vamos analisar a **importância dos testes e auditorias** para cada um dos casos.

Caso Knight Capital

Tipo de Teste	Como Ajudaria no Caso Knight Capital
Testes de Integração	Detectariam conflitos entre o novo código e o legado (“Power Peg”) antes de ir para produção.
Testes de Regressão	Garantiriam que funcionalidades antigas não fossem ativadas por engano.
Testes em Ambiente Simulado	Permitiriam simular cenários de mercado e identificar comportamentos inesperados sob alta carga.
Testes Automatizados	Alertariam rapidamente sobre disparos anômalos de ordens.

Caso Therac-25

Tipo de Teste	Como Ajudaria no Caso Therac-25
Testes de Integração	Verificariam a interação entre o software e os sistemas de hardware de segurança, evitando falhas.

Tipo de Teste	Como Ajudaria no Caso Therac-25
Testes de Regressão	Garantiriam que novas mudanças não afetassem a segurança, especialmente em relação aos limites de radiação.
Testes de Performance	Avaliariam a estabilidade do sistema sob condições de carga elevada e validação de segurança.

5. Papel da Gestão de Requisitos

Caso Therac-25

Etapa da Gestão de Requisitos	Exemplo Aplicado ao Therac-25	Benefício
Elicitação de Requisitos	Identificar todas as condições de segurança para evitar exposição excessiva de radiação.	Garantia de que todos os cenários críticos sejam considerados desde o início, reduzindo riscos de falhas fatais.
Documentação Formal	Especificando requisitos claros, rastreáveis e testáveis.	Facilita a verificação e validação do software antes da produção, evitando falhas.
Análise de Consistência e Completude	Revisão detalhada para detectar lacunas em requisitos críticos, como falhas de concorrência.	Minimiza o risco de execução de funções inesperadas ou perigosas, como a irradiação excessiva de pacientes.
Rastreabilidade	Mapear cada requisito crítico para código, testes e validações físicas.	Permite verificar que cada requisito foi implementado corretamente e testado em todas as fases.
Gestão de Mudanças	Controle rigoroso sobre qualquer alteração de requisitos durante desenvolvimento.	Evita introdução de falhas durante atualizações ou reuso de código legado.