UNIVAS
UNIVERSIDADE DO VALE DO SAPUCAÍ

Gestão da Qualidade de Software

Métricas de Qualidade de Código

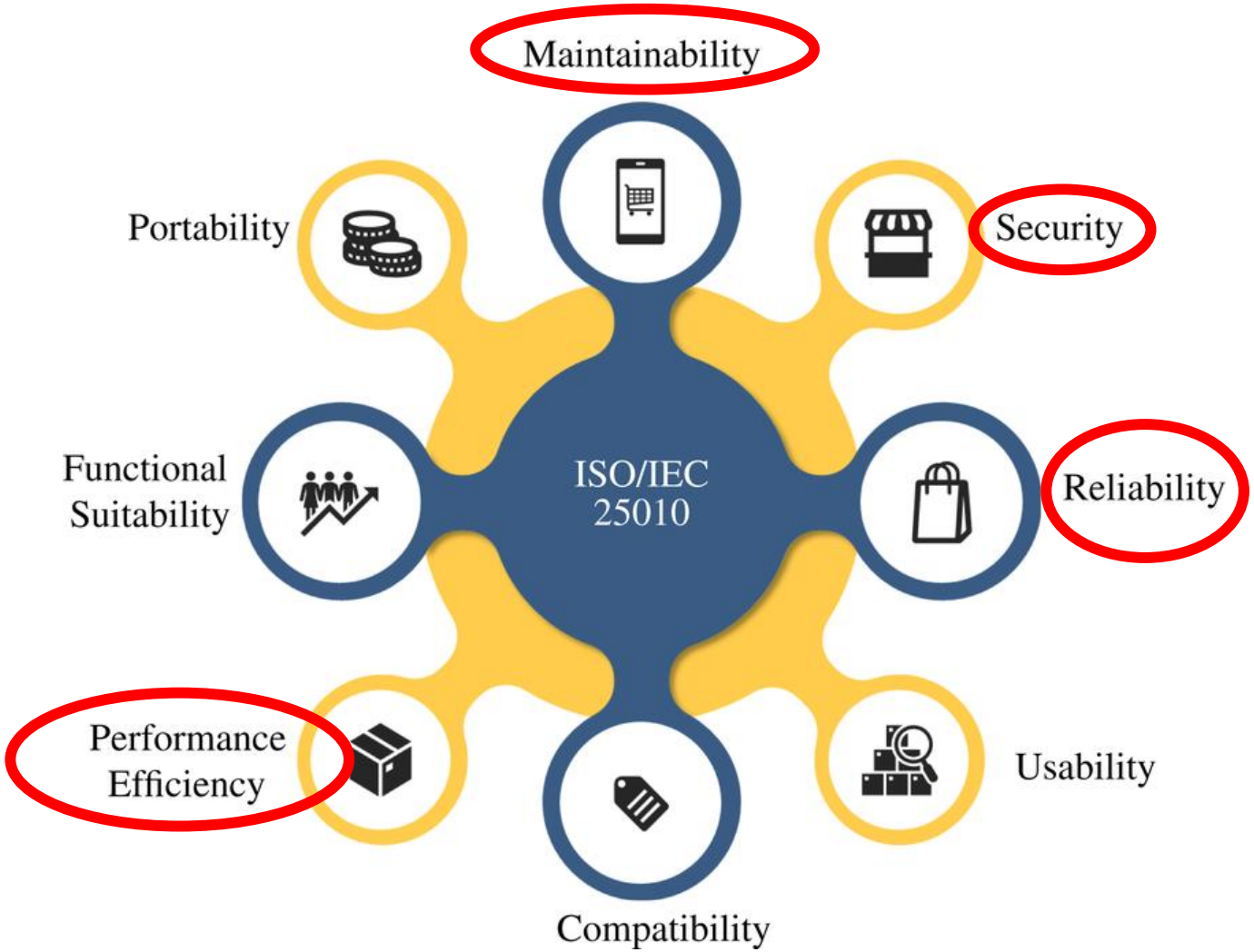
Prof. Flávio Belizário da Silva Mota
Universidade do Vale do Sapucaí – UNIVAS
Sistemas de Informação

Métricas de Qualidade de Código

- Na última aula vimos:
  - Métricas de produto, processo e projeto.
  - Critérios para escolher boas métricas.
  - Paradigma GQM como forma de alinhar métricas a objetivos.
- Na aula de hoje, vamos aprofundar em métricas de produto aplicadas ao **código-fonte**, que refletem a qualidade interna.
- Qualidade de código é a base para que o **software seja sustentável ao longo do tempo**.
- Essas métricas são fundamentais para garantir **qualidade e reduzir custos de manutenção.**

Qualidade de código

- Qualidade de código significa que o software é:
  - Fácil de manter (corrigir, evoluir, adaptar)
  - Claro e compreensível (para novos desenvolvedores)
  - Facilmente testável (automatização e confiabilidade)



Dívida técnica

- Dívida técnica é o “custo futuro” de se escolher soluções **rápidas** em detrimento das **corretas**.
- Exemplos:
  - Código duplicado
  - Falta de testes automatizados
  - Estrutura confusa e pouco legível

As **métricas de código** ajudam a monitorar e reduzir a dívida técnica.

Code Smells

- *Code smells* são indícios de problemas no código que comprometem qualidade.
  Exemplos:
    - Funções muito longas
    - Classes 'Deus' (fazem tudo)
    - Variáveis não utilizadas
    - Código duplicado

Complexidade ciclomática

- O cálculo de complexidade ciclomática leva em conta quantos desvios uma função possui.
- Cada comando *if*, *switch*, *for*, *while*, é considerado um desvio do fluxo principal do código.
- Toda função possui como valor padrão 1, já que pelo menos um caminho é possível para o código.
- Conforme são inseridos os desvios este número aumenta.

*"Cada método **deve ter uma única responsabilidade**, portanto devemos sempre buscar manter este indicador baixo, garantindo a manutenibilidade de nosso código"* (Código Limpo)

Índice de Manutenibilidade

- Este índice avalia a facilidade de manutenção do código em uma escala de 0 a 100.
- Um valor mais alto (normalmente acima de 85) indica melhor manutenibilidade.
- Ele leva em consideração fatores como extensibilidade, número de linhas de código, complexidade ciclomática e medidas de complexidade.
- Ele é calculado usando uma fórmula que combina diversas métricas, incluindo **volume de código**, **complexidade** e **comentários**. Isso ajuda a identificar áreas onde a refatoração pode reduzir a complexidade.

Duplicação de código

- Mede a quantidade de código duplicado em um software.
- Altos níveis de duplicação podem dificultar a manutenção e levar a mais bugs, pois há a possibilidade de implementações divergentes da mesma funcionalidade.
- Detectado usando ferramentas que procuram blocos de código idênticos ou semelhantes.
- Reduzir a duplicação pode melhorar significativamente a qualidade do código, garantindo a consistência e reduzindo potenciais pontos de falha.

Taxa de aprovação em testes unitários

- Esta métrica indica a porcentagem de testes unitários aprovados durante uma execução de teste.
- Ela reflete a **confiabilidade** do código sob **condições de teste**.
- Taxas baixas podem indicar problemas na lógica do código ou lacunas na cobertura do teste, o que exige uma investigação mais aprofundada.

Rotatividade de código

- A rotatividade de código mede a frequência com que o código é adicionado, modificado ou excluído em um **repositório**.
- Altas taxas de rotatividade podem indicar **instabilidade ou indecisão na equipe de desenvolvimento**.
- Acompanhar a rotatividade de código ao longo do tempo ajuda a identificar áreas do código que mudam com frequência e podem se beneficiar de refatoração ou testes adicionais de design.

Tempo médio de revisão de código

- Esta é uma medida **quantitativa** do tempo médio gasto em revisões de código.
- Tempos de revisão **longos** podem indicar **complexidade** ou **falta de clareza**, enquanto tempos muito **curtos** podem sugerir **rigor insuficiente**.
- Monitorar esta métrica pode ajudar as equipes a entender e aprimorar seus processos de revisão, buscando revisões completas, porém eficientes, para manter o ritmo sem sacrificar a qualidade.

Exemplo

Código A

```python
def calcular_desconto(valor):
    if valor > 1000:
        return valor * 0.90  # 10% de desconto
    return valor
```

- Qual das duas versões é mais fácil de compreender e alterar?
- Qual teria maior risco de introduzir erros ao ser modificada?

Código B

```python
def calcular_desconto(valor, cliente, promocao):
    if cliente == "VIP":
        if valor > 2000:
            if promocao:
                return valor * 0.75  # VIP + promoção
            else:
                return valor * 0.80
        else:
            return valor * 0.85
    elif cliente == "Novo":
        if promocao and valor > 1000:
            return valor * 0.85
        elif promocao:
            return valor * 0.90
        else:
            return valor
    else:
        if valor > 500:
            return valor * 0.95
        return valor
```

Como times podem analisar a qualidade de código?

- **Verificações de Qualidade Automatizadas**
  - Envolvem o uso de **ferramentas de software** para analisar automaticamente o código em busca de problemas potenciais
- **Revisões Manuais de Código**
  - São uma análise sistemática do **código** por desenvolvedores que **não escreveram originalmente o código**, com o objetivo de encontrar bugs, garantir a aderência aos padrões de codificação e melhorar a qualidade do código

Como times podem analisar a qualidade de código?

- **Análise Estática**
  - Examina o código sem executar o programa. Nenhum tipo de teste em tempo real é realizado durante o processo, portanto, **não há clareza sobre a funcionalidade real do programa**.
  - O código é testado em relação a determinadas diretrizes de codificação para garantir que atenda a determinados padrões e que o programa esteja funcionando conforme o esperado.

Ferramentas para medir

- **SonarQube**: análise contínua de qualidade de código
- **JMeter**: desempenho de software (cargas de trabalho)
- **Selenium**: automatização de testes
- **OWASP ZAP**: teste de segurança