AULA 09: Monitoramento da Qualidade Interna: Métricas de código-fonte.

Professora: **Rosana T. Vaccare Braga**

Agenda

- Conceito de monitoramento da qualidade interna
- Introdução às métricas de código-fonte
- Uso das medições de um software de sistema
- Métricas de tamanho e complexidade
- Métricas de qualidade
- Métricas orientadas a objeto
- Como coletar métricas de código-fonte?

Conceito de monitoramento da qualidade interna

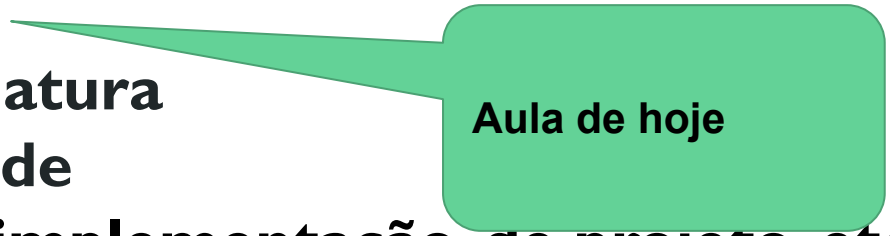
- **O monitoramento da qualidade interna de software consiste na avaliação sistemática dos atributos e métricas internas do código, visando identificar possíveis falhas, inconsistências ou oportunidades de melhoria na estrutura do software.**
- **Essa prática é fundamental para garantir a qualidade e a confiabilidade do produto final, bem como para aprimorar o processo de desenvolvimento de software de forma contínua.**

Conceito de monitoramento da qualidade interna

- **Ao monitorar a qualidade interna:**
  - **é possível identificar problemas precocemente, antes que eles se tornem mais difíceis e onerosos de serem corrigidos**
  - **as medidas obtidas por meio do monitoramento da qualidade interna podem auxiliar na definição de objetivos claros de qualidade, no planejamento de melhorias no código-fonte e no acompanhamento dos resultados alcançados.**

Conceito de monitoramento da qualidade interna

- **Algumas formas de monitorar a qualidade interna:**
  - **métricas de software**
  - **convenção de nomenclatura**
  - **estratégias de clean code**
  - **adoção de padrões de implementação, de projeto, etc.**
  - **e outros**



Introdução às métricas de código-fonte

- **Conjunto de técnicas e ferramentas que permitem medir a qualidade e a eficiência do código escrito por desenvolvedores de software.**

- **São utilizadas para avaliar a manutenibilidade, a legibilidade, a complexidade e outras características do código que impactam diretamente na qualidade do software.**

Introdução às métricas de código-fonte

- **Métricas são importantes porque permitem aos desenvolvedores e gerentes de projeto avaliar a qualidade do código em tempo real, identificar possíveis problemas e tomar decisões embasadas em dados para melhorar a qualidade do software.**

Métrica x medição x medida

Métrica: técnica utilizada para medir algo
Medição: ato de aplicar uma métrica
Medida: valor atribuído durante a medição

Usei a fita métrica para fazer a medição de minha cintura e o valor medido foi 80 cm.

Introdução às métricas de código-fonte

- **Métricas podem ser utilizadas para rastrear a evolução do código ao longo do tempo e para comparar a qualidade do código entre diferentes projetos ou equipes.**

- **São uma ferramenta essencial para garantir que o software seja fácil de manter e evoluir ao longo do tempo.**

Software A v1.0

10000 linhas de código
500 linhas de comentário
50 classes
430 métodos

Software A v2.0

10950 linhas de código
1300 linhas de comentário
53 classes
485 métodos

Introdução às métricas de código-fonte

- **Medição de software: preocupa-se com a derivação de um valor numérico ou o perfil para um atributo de um componente de software, sistema ou processo.**
  - **Comparando esses valores entre si e com os padrões que se aplicam a toda a organização, você pode ser capaz de tirar conclusões sobre a qualidade do software ou avaliar a eficácia dos métodos, das ferramentas e dos processos de software.**

Exemplos de métricas

- **índice Fog [url], é uma medida da legibilidade de um trecho de texto escrito;**

Índice Fog: https://corporatefinanceinstitute.com/resources/management/fog-index/

Exemplos de métricas de código-fonte

- **tamanho de um produto em linhas de código;**
- **número de defeitos relatados em um produto de software entregue;**
- **número de pessoas/dia requerido para desenvolver um componente de sistema.**

Introdução às métricas de código-fonte

- **Exemplo:**
  - **uma organização tem a intenção de introduzir uma nova ferramenta de teste de software.**
  - **Antes de introduzir a ferramenta, em um determinado momento você registra o número de defeitos de software descobertos.**
    - **Essa é uma baseline de avaliação da eficácia da ferramenta.**
  - **Depois de algum tempo usando a ferramenta, esse processo é repetido.**
  - **Se mais defeitos forem encontrados durante o mesmo período, depois que a ferramenta foi introduzida, você pode decidir se ela fornece suporte útil para o processo de validação de software.**

EXEMPLO

Antes da introdução da nova ferramenta de teste

5 defeitos por 1000 linhas de código num período de 1 semana

4 semanas após a introdução da nova ferramenta de teste

15 defeitos por 1000 linhas de código num período de 1 semana

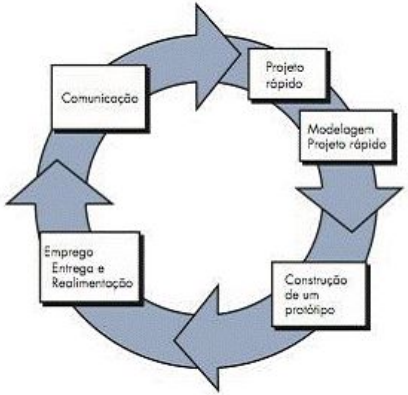
Decisão: ferramenta encontrou mais erros, vamos adotá-la!

Introdução às métricas de código-fonte

- **O objetivo a longo prazo de medição de software é usá-la no lugar de revisões para julgar a qualidade de um software.**
  - **Usando a medição de software, um sistema poderia, idealmente, ser avaliado usando uma variedade de métricas e, a partir dessa medição, deduzir um valor para a qualidade do sistema.**
  - **Se o software atingir o limiar de qualidade requerido, ele poderia ser aprovado sem revisão.**
  - **Quando apropriado, as ferramentas de medição também podem realçar áreas do software que poderiam ser melhoradas.**
  - **No entanto, estamos ainda longe dessa situação ideal e não há sinal de que as avaliações automatizadas de qualidade venham a se tornar realidade em um futuro próximo.**

Introdução às métricas de código-fonte

- **As métricas de software podem ser:**
  - **métricas de controle: apoiam os processos de gerenciamento. Geralmente são associadas com os processos de software.**
  - **métricas de previsão: ajudam a prever as características do software.**

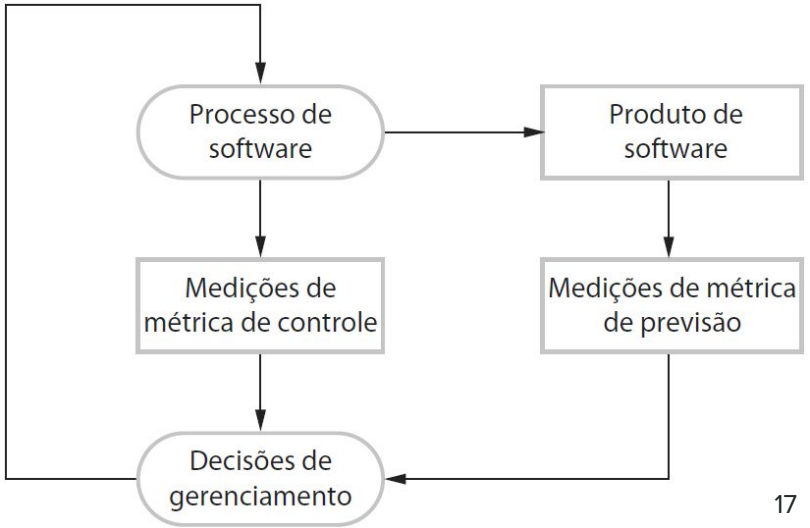


Introdução às métricas de código-fonte

- **Exemplos de métricas de controle ou de processos são o esforço médio e o tempo necessário para reparar os defeitos relatados.**
- **Métricas de previsão são associadas com o software em si e, por vezes, são conhecidas como 'métricas de produto'.**
  - **Exemplos: a complexidade ciclomática de um módulo, o comprimento médio dos identificadores em um programa e o número de atributos e operações associadas com as classes de objeto em um projeto.**

Introdução às métricas de código-fonte

- **As métricas de controle e de previsão podem influenciar a tomada de decisão de gerenciamento.**
  - **Os gerentes usam métricas de controle para decidir se devem ser feitas alterações no processo;**
  - **As métricas de previsão são usadas para ajudar a estimar o esforço necessário para fazer as alterações no software.**



Medições de previsão e de controle - Sommerville 9ª edição

Introdução às métricas de código-fonte

Segundo o ISO/IEC 9126-1 (2001), as métricas de software podem ser classificadas em:

- **Métricas internas: baseadas em medidas estáticas obtidas pela análise do código, exemplos incluem simplicidade, concisão, coesão, clareza, baixo acoplamento e generalidade.**
- **Métricas externas: aplicáveis ao software em execução, exemplos incluem correção, usabilidade, eficiência e robustez.**
- **Métricas de qualidade em uso: coletadas durante o uso do software pelos usuários finais em condições reais.**

Uso das medições de um software de sistema

- **Existem duas maneiras para o uso das medições de um software de sistema:**

  1. Para atribuir um valor aos atributos de qualidade de sistema. Ao medir as características dos componentes de sistema, bem como sua complexidade ciclomática e, em seguida, agregar essas medições, você pode avaliar os atributos de qualidade do sistema, como a manutenibilidade.

  2. Para identificar os componentes de sistema cuja qualidade não atingiu o padrão. As medições podem identificar componentes individuais com características que se desviem da norma. Por exemplo, você pode medir componentes para descobrir aqueles com a mais alta complexidade. Esses são mais passíveis de conter bugs porque a complexidade os torna mais difíceis de entender.

Uso das medições de um software de sistema

- **Infelizmente, é difícil fazer medições diretas de muitos dos atributos de qualidade de software. Os atributos de qualidade como manutenibilidade, compreensibilidade e usabilidade são atributos externos relacionados com os desenvolvedores e usuários que experimentam o software.**
- **Eles são afetados por fatores subjetivos, como a experiência e a educação do usuário e, portanto, não podem ser medidos objetivamente.**
  - **Para fazer um julgamento sobre esses atributos, você deve medir alguns atributos internos do software (como tamanho, complexidade etc.) e assumir que estão relacionados com as características de qualidade com as quais você se preocupa.**

Uso das medições de um software de sistema

De acordo com Sommerville, existem várias razões pelas quais isso é difícil:

1. **É impossível quantificar o retorno sobre o investimento da introdução de um programa de métricas em organizações.**
2. **Não existe um padrão para as métricas de software ou processos padronizados para medição e análise.**
3. **Em muitas empresas, os processos de software não são padronizados e são mal definidos e mal controlados.**
4. **Grande parte da pesquisa sobre medição e métricas de software centra-se nas métricas baseadas em códigos e processos de desenvolvimento dirigidos a planos.**
5. **A introdução da medição acrescenta overhead aos processos.**

Uso das medições de um software de sistema

- **De acordo com Sommerville, mesmo quando é possível fazer medições objetivas e tirar conclusões a partir delas, isso pode não convencer os tomadores de decisões necessariamente.**
- **Em vez disso, as tomadas de decisões muitas vezes são influenciadas por fatores subjetivos, como a novidade ou a extensão em que as técnicas são de interesse para os profissionais.**

Uso das medições de um software de sistema

- **As métricas devem estar associadas a uma escala de medida que dê sentido ao valor obtido em seu cálculo.**
- **Antes de fazer comparações entre valores de métricas, é necessário conhecer o modelo de dados a que pertencem.**

Uso das medições de um software de sistema

Os valores podem pertencer a um desses tipos de escala estatística (Conte et al., 1986):

- **Nominal: os valores não possuem ordem nem magnitude. Essa escala é utilizada para separar categorias, onde os valores são apenas rótulos que podem ser comparados por igualdade. Exemplos de variáveis nominais incluem estado civil, código postal e código de barras.**
- **Ordinal: Os valores podem ser comparados segundo uma ordem, mas não existe uma definição clara da magnitude da diferença entre eles. Um exemplo de variável ordinal é o tamanho de roupa, onde os valores são P, M, G e GG, mas não é possível determinar a magnitude exata da diferença entre o tamanho P e o tamanho M, por exemplo.**

Uso das medições de um software de sistema

Os valores podem pertencer a um desses tipos de escala estatística (Conte et al., 1986):

- **Intervalar: há ordem e tamanho dos intervalos que separam os valores, porém proporções não são aplicáveis, já que não há uma unidade natural ou zero absoluto. Um exemplo seria a escala Celsius de temperatura, onde não é possível afirmar que 40 graus é o dobro de calor que 20 graus.**
- **Racional: possui ordem, magnitude, zero absoluto e unidade natural. Exemplos incluem preço, idade e distância.**

Métricas de tamanho e complexidade

- **As métricas de tamanho e complexidade são fundamentais para a medição e avaliação da qualidade de software.**
  - **Desde os primórdios da computação, o tamanho do software tem sido uma medida importante, pois é um requisito de padrões de qualidade e certificações de processos de desenvolvimento, como o Capability Maturity Model Integration (CMMI).**
  - **As métricas de tamanho são usadas por modelos de estimativas de custo e esforço, como o Cocomo e o Cosysmo.**

Métricas de tamanho e complexidade

- **Por sua vez, a complexidade do software está relacionada ao número de interações entre diferentes elementos do sistema.**
  - **Quanto mais interações existirem, maior será a probabilidade de que uma mudança em uma parte do sistema afete outras partes, exigindo mais mudanças ou introduzindo falhas inesperadas.**
  - **Quanto mais entidades tiver o sistema, maior será o número possível de interações entre elas. Assim, sistemas maiores são mais suscetíveis a maior complexidade, e métricas de complexidade devem ser usadas em conjunto com métricas de tamanho para uma avaliação mais completa da qualidade do software.**

Métricas de tamanho e complexidade

- **LOC, ou Linhas de Código, é a métrica mais antiga e conhecida para medir o código-fonte.**
  - **Embora tenha sido fácil de definir em linguagens mais antigas, como FORTRAN e assembler, as linguagens modernas permitem uma variedade maior de estilos de programação, o que torna o cálculo das linhas de código mais complexo. Por isso, foram criadas variações dessa métrica.**

Métricas de tamanho e complexidade

- **Linhas de código (LOC ou SLOC - Source Lines Of Code):**
  - **A métrica mais utilizada para medir o tamanho de um software, consiste em contabilizar todas as linhas que contêm código e excluir da contagem comentários e linhas em branco.**
  - **Embora seja uma métrica amplamente conhecida e utilizada, ela tem suas limitações, especialmente em linguagens modernas, que permitem diferentes estilos de programação e podem produzir diferentes resultados de contagem.**

Métricas de tamanho e complexidade

- **Linhas físicas de código**:
  - **conta o número de linhas em cada arquivo de código-fonte, independentemente do conteúdo. Portanto, linhas em branco e comentários são incluídos na contagem, e cada linha é contada apenas uma vez, mesmo que contenha mais de uma instrução.**
  - **Essa métrica é raramente usada devido à sua falta de robustez, uma vez que varia muito com a formatação e estilo de codificação.**

Métricas de tamanho e complexidade

- **Linhas lógicas de código (LLOC - Logical Lines Of Code):**
  - **São uma métrica que contabiliza o número de instruções em um código. Geralmente é calculada contando o número de terminadores, como pontos-e-vírgulas.**
  - **Dessa forma, uma linha que contém apenas um teste condicional ou declaração de função não é considerada, enquanto uma linha que contém duas instruções é contada duas vezes.**
  - **A LLOC é uma métrica mais precisa do que a contagem de linhas físicas de código, pois leva em consideração a estrutura lógica do código e ignora a formatação e o estilo de codificação.**

Métricas de tamanho e complexidade

- **Linhas efetivas de código (ELOC - Effective Lines Of Code):**
  - **contabilizam todas as linhas de código que possuem instruções, excluindo aquelas que contêm apenas delimitadores, como chaves, parênteses, aspas, begin e end.**
  - **essa métrica é mais precisa que a SLOC para estimar esforço, pois é menos suscetível a variações de estilo, mas é menos utilizada por poucas ferramentas oferecerem suporte.**

Métricas de tamanho e complexidade

- **Resumindo: a métrica LOC é simples e fácil de calcular, mas é sensível a diferenças na linguagem de programação, o que pode levar a problemas na comparação de projetos em linguagens diferentes.**
- **Fatores de normalização ou comparação em escala logarítmica podem ser usados para lidar com essas diferenças, mas projetos multilinguagens ainda apresentam desafios na definição de medidas como erros por linha de código.**

Métricas de tamanho e complexidade

- **Pontos de função (FP) (Function Points) são uma métrica que busca representar a funcionalidade entregue aos usuários do sistema.**
- **Existem vários padrões para análise ou obtenção automática de FP, que consideram o número de entradas do usuário, as consultas, as saídas e os principais arquivos.**
- **Essa métrica foi proposta como uma alternativa às linhas de código para medir a produtividade sem incentivar prolixidade e estimar custos com base apenas nos requisitos. O custo de 1 ponto pode ser estimado com base em experiências prévias com outros projetos.**
- **A métrica foi estendida posteriormente para Pontos por Caso de Uso**

Métricas de tamanho e complexidade

- **As Métricas de Halstead constituem um conjunto de métricas fundamentado na teoria da informação, conforme proposto por Halstead em 1977. Tais métricas são aplicáveis a diversos elementos do software e objetivam estimar o esforço, tempo de desenvolvimento e até mesmo o número esperado de bugs, utilizando como base somente a quantidade de operadores e operandos presentes.**

https://acervolima.com/engenharia-de-software-metricas-de-software-da-halstead/

Métricas de tamanho e complexidade

- **A complexidade ciclomática é uma métrica baseada no grafo de controle de fluxo do programa e representa o número de ciclos independentes nesse grafo.**

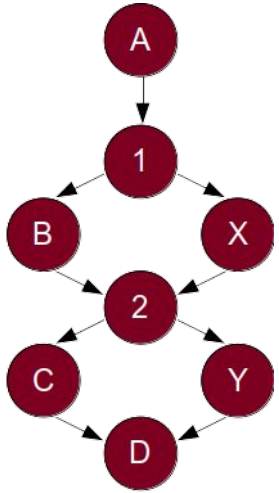
- **Essa definição é equivalente ao número de estruturas condicionais mais um, o que é facilmente computável.**

- **A métrica é também chamada de complexidade condicional, pois é obtida contando-se o número de estruturas condicionais.**

Métricas de tamanho e complexidade

- **A complexidade ciclomática é calculada a partir de um grafo de fluxo de controle do programa, que representa as possíveis rotas de execução do código. Cada nó do grafo representa um bloco básico de código, e cada aresta representa uma transição de controle entre os blocos.**
- **A complexidade ciclomática é definida como o número de regiões ou ciclos independentes no grafo de fluxo de controle. Isso significa que quanto mais caminhos independentes existirem no código, maior será sua complexidade ciclomática e mais difícil será de entender e manter o código.**

```
print("A")
if (condition1)
  print("X")
else
  print ("B")
if (condition2)
  print("Y")
else
  print ("C")
print("D")
```



Uma alta complexidade ciclomática pode indicar um código com muitos desvios condicionais, loops e funções com muitas linhas de código. Isso pode tornar o código mais difícil de testar e depurar, além de aumentar o risco de erros e problemas de desempenho.

Métricas de qualidade

- **São usadas para medir diferentes aspectos da qualidade interna do código e auxiliar na identificação de possíveis problemas.**
- **Principais métricas:**
  - **Cobertura de testes**
  - **Duplicação de código**
  - **Taxa de defeitos**
  - **Conformidade com padrões**
- **Essas métricas podem ser coletadas automaticamente por ferramentas de análise de qualidade de código, como SonarQube, Checkstyle e PMD, permitindo que os desenvolvedores monitorem a qualidade interna do software e identifiquem possíveis problemas.**

Métricas de qualidade - Duplicação de código

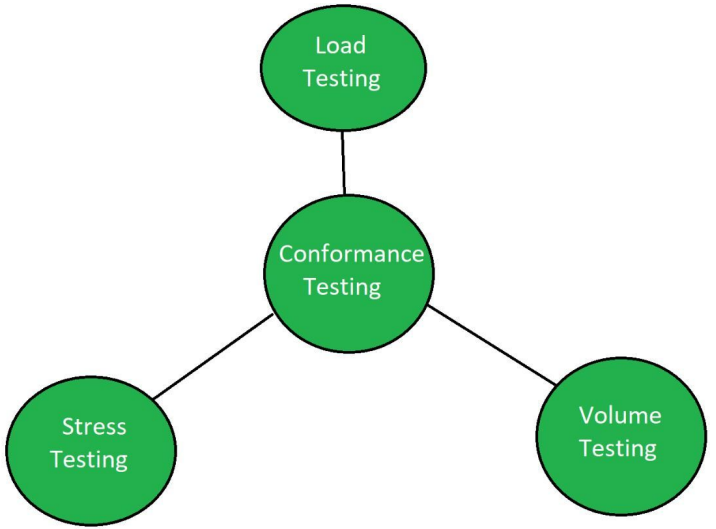
- **Métrica que mede a quantidade de código repetido em um projeto. Quando há uma grande quantidade de código duplicado, é maior a chance de inconsistências e erros no software, já que uma mudança em uma parte do código pode afetar outras partes repetidas. Por isso, a redução da duplicação de código é importante para aumentar a eficiência e a qualidade do software, além de tornar a manutenção mais fácil e menos propensa a erros.**

Métricas de qualidade - Taxa de defeitos

- **Mede a quantidade de defeitos encontrados em um determinado tamanho de código. Ela é geralmente expressa como o número de defeitos por mil linhas de código ou por função. Quanto menor a taxa de defeitos, mais confiável é o software.**
  - **Uma baixa taxa de defeitos indica que o software é menos propenso a falhas e erros, o que significa que é mais confiável e seguro para uso.**
  - **A taxa de defeitos também pode ser usada para monitorar a qualidade do software ao longo do tempo e para identificar áreas problemáticas do código que precisam de mais atenção e trabalho de correção.**
  - **É importante ressaltar que a taxa de defeitos deve ser utilizada em conjunto com outras métricas para avaliar a qualidade do software como um todo.**

Métricas de qualidade - Conformidade com padrões

- **Métrica de qualidade de software que mede a adesão do código às regras estabelecidas.**
- **A adesão aos padrões ajuda a garantir a consistência do código em todo o projeto, facilitando a compreensão e manutenção do software. Além disso, a adesão a padrões de codificação pode ajudar a evitar vulnerabilidades de segurança, já que muitas diretrizes de codificação incluem práticas recomendadas para evitar vulnerabilidades conhecidas.**



Métricas de qualidade - Cobertura de testes

- **A cobertura de testes é uma métrica utilizada para avaliar a efetividade dos testes automatizados de um software. Essa métrica mede a porcentagem do código que é executado pelos testes automatizados. Quanto maior a cobertura de testes, ou seja, quanto mais código é testado, maior é a confiabilidade do software, uma vez que há uma maior garantia de que todas as funcionalidades estão funcionando corretamente e não há falhas não detectadas.**

Alta cobertura de testes não garante software sem problemas ou código organizado. Outras métricas devem ser usadas para avaliar a qualidade geral.

Métricas orientadas a objeto

- **Métricas de código-fonte para software orientado a objetos diferem das métricas de código para outros paradigmas, pois usam entidades em vez de algoritmos como componentes fundamentais.**
  - **Além de tamanho e complexidade, tais métricas podem medir o uso de herança, polimorfismo, encapsulamento e interdependência entre as entidades.**
  - **As métricas mais comuns em ferramentas de análise de código-fonte são as dos grupos:**
    - **CK (Chidamber e Kemerer, 1994) e**
    - **MOOD (Abreu e Carapuça, 1994).**

Grupo CK - Chidamber e Kemerer

- **A profundidade da árvore de herança (DIT - Depth of Inheritance Tree) é a distância máxima de uma classe até a raiz na árvore de herança.**
- **Em linguagens com herança múltipla, a definição usa o termo "distância máxima", enquanto em linguagens com herança simples é o número de ancestrais.**
- **Mais ancestrais aumentam a chance de herdar atributos e métodos, mas valores altos indicam uma hierarquia complexa e valores baixos indicam subutilização da herança.**

Grupo CK - Chidamber e Kemerer

- **A métrica NOC (Number Of Children), ou número de filhos, é definida como a contagem de subclasses diretas da classe em análise.**
  - **Alterações em uma classe podem impactar todas as suas subclasses, portanto, quanto mais subclasses uma classe tiver, maior será sua influência no sistema e mais importante será que ela seja bem testada.**
  - **O uso excessivo de subclasses pode indicar um uso inadequado do mecanismo de herança.**
  - **A métrica NOC pode ser utilizada em conjunto com DIT para avaliar o uso da herança em um sistema.**

Grupo CK - Chidamber e Kemerer

- **O acoplamento entre objetos, ou CBO (Coupling Between Objects), é uma medida do número de classes acopladas a uma determinada classe, ou seja, o número de classes que são acessadas pela classe em questão (conexões eferentes) somado ao número de classes do sistema que acessam essa classe (conexões aferentes).**
  - **Uma classe acessa outra quando invoca seus métodos, lê ou modifica seus atributos.**
  - **Quanto mais independente uma classe for, mais fácil será reutilizá-la e menos arriscado será modificá-la.**
  - **Classes com maior acoplamento exigem mais rigor em testes, pois mais partes do sistema dependem delas.**

Grupo CK - Chidamber e Kemerer

- **Acoplamento entre objetos (CBO)**

Grupo CK - Chidamber e Kemerer

- **RFC (do inglês Response For a Class) é uma métrica que representa o número total de métodos de uma classe somado ao número de métodos que a classe invoca.**
  - **Essencialmente, é o número de métodos que podem ser executados como resultado de uma mensagem enviada à classe.**
  - **Classes com valores altos de RFC tendem a ter muitos métodos e/ou chamar muitos métodos. Isso torna mais difícil compreender e testar o comportamento da classe.**

Grupo CK - Chidamber e Kemerer

- **A falta de coesão em métodos é uma medida da falta de coesão de uma classe. LCOM4 (Lack of Cohesion in Methods) é a sigla comumente utilizada para a versão 4 desta métrica.**
  - **Definimos M = {M1, . . . , Mn} como o conjunto dos métodos da classe analisada. Dois métodos Mi e Mj são considerados coesos se ambos acessam pelo menos um mesmo atributo da classe ou se Mi chama ou é chamado por Mj.**
  - **LCOM4 é o tamanho da partição formada pela separação de M subconjuntos coesos.**
  - **Uma classe completamente coesa tem LCOM4 igual a 1, enquanto valores maiores que 1 indicam que a classe pode violar o princípio de responsabilidade única (SRP) e possivelmente deve ser quebrada em duas ou mais classes coesas.**

Grupo MOOD - Metrics for Object Oriented Design

- **Métricas do grupo MOOD são adimensionais e independentes de tamanho e linguagem de programação, variando de 0 a 1 e representando porcentagens para facilitar a comparação entre projetos.**

Grupo MOOD - Metrics for Object Oriented Design

- **Acoplamento:** mede o grau de interdependência entre os módulos ou componentes de um sistema.
  - Segundo Sommerville, é uma medida de quanto acoplamento existe. Um valor alto significa que as classes são altamente dependentes e, portanto, é mais provável que a mudança em uma classe afete outras classes do programa.

Grupo MOOD - Metrics for Object Oriented Design

- **Coesão:** mede o grau em que as responsabilidades de um módulo ou componente estão relacionadas e coesas.
  - De acordo com Sommerville, a coesão é calculada considerando os pares de métodos em uma classe. A coesão é a diferença entre o número de pares de métodos sem atributos compartilhados e o número de pares de métodos com atributos compartilhados. O valor dessa métrica tem sido amplamente discutido, e ele existe em diversas variações. Não está claro se realmente adiciona qualquer informação útil além das que já são fornecidas por outras métricas.

Grupo MOOD - Metrics for Object Oriented Design

- **O fator de acoplamento, ou COF (Coupling Factor em inglês), é uma métrica amplamente utilizada para avaliar o nível de acoplamento de um sistema. Seu valor é determinado por meio da seguinte fórmula:**

$$COF = \frac{\sum_{i=1}^{n} AC(C_i)}{n^2 - n}$$

Onde n é o número de classes e AC(Ci) é o número de conexões aferentes da classe Ci . O numerador é o total de ligações entre as classes e o denominador é o total possível de ligações. Um software fortemente conectado tem baixo grau de independência entre os módulos, sendo mais difícil entender, testar e modificar.

Grupo MOOD - Metrics for Object Oriented Design

As outras métricas desse grupo, menos populares, são:

- **Fator de ocultação de métodos (MHF - Method Hiding Factor), a porcentagem de métodos não-públicos.**
- **Fator de ocultação de atributos (AHF - Attribute Hiding Factor), a porcentagem de atributos não-públicos, que junto com MHF mede encapsulamento.**
- **Fator de herança de métodos (MIF - Method Inheritance Factor), a porcentagem de métodos herdados.**
- **Fator de herança de atributos (AIF - Attribute Inheritance Factor), a porcentagem de atributos herdados, que junto com MIF mede o impacto da herança.**
- **Fator de polimorfismo (PF - Polymorphism Factor), que mede o uso de polimorfismo.**

Como coletar métricas de código-fonte?

- As métricas de código-fonte podem ser coletadas de várias maneiras, incluindo ferramentas automatizadas e análise manual.
- Ferramentas de análise estática de código-fonte, como o SonarQube, podem ser usadas para coletar métricas de forma automatizada, fornecendo relatórios detalhados sobre a qualidade do código e a conformidade com as boas práticas de programação.
  - Essas ferramentas geralmente utilizam um conjunto de regras predefinidas e padrões de codificação para avaliar o código-fonte e gerar métricas como complexidade ciclomática, número de linhas de código, cobertura de teste, entre outras.

Como coletar métricas de código-fonte?

- **Coleta de métricas de forma manual:**
  - **revisões de código por pares ou inspeções de código: revisores avaliam o código-fonte de forma crítica, identificando potenciais problemas e pontos de melhoria, enquanto coletam informações relevantes para gerar métricas, como o tempo necessário para corrigir problemas ou o número de erros encontrados em cada revisão.**
- **Independentemente da abordagem escolhida, a coleta de métricas de código-fonte deve ser um processo contínuo e integrado ao processo de desenvolvimento de software.**
  - **Isso permite a identificação precoce de problemas e a tomada de decisões baseadas em dados para melhorar a qualidade do código e aumentar a eficiência do processo de desenvolvimento.**

Próxima aula

- Convenção de nomenclatura
- Estratégias de código limpo
  - Princípios de código limpo
  - Refatoração de código
  - **Análise estática de código**
  - **Práticas de desenvolvimento de equipe**