Análise de Código com a Ferramenta PMD

"Programming Mistake Detector"

Diogo Conforti Vaz Bellini
João Paulo Morais Rangel
Leonardo Poloni Berti Morikio

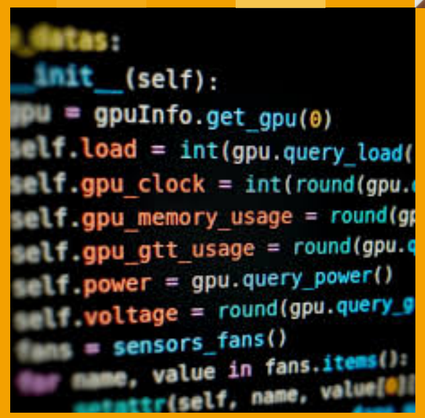
Fontes

- PMD Official Site
  - What does PMD mean
  - Logo
  - IDE Plugins
    - IntelliJ (jetbrains)
  - Intallation and basic CLI usage
    - Github
    - PMD CLI reference
  - Making Rulesets
  - Configuring Rule
  - Writing rules
    - XPath (w3schools)
  - Java Rules
  - Maven PMD Puglin

Tópicos

- Introdução Geral
- Principais vantagens da análise estática
- Introdução ao PMD
- Funcionamento do PMD
- Análise de árvore sintática
- Regras e rulesets PMD
- Ruleset padrão (JAVA)
- Como configurar e executar o PMD
- Exemplo de Código com Problemas
- Análise do Código com PMD
- Personalização de rulesets e regras
- Integração do PMD com o processo de desenvolvimento
- Exemplo de integração do PMD com Maven
- Integração do PMD com Maven
- Vantagens e Limitações

Introdução Geral



- Análise estática do código
- Detecção precoce de erros, padronização, manutenção facilitada, segurança.
- PMD, Checkstyle, SonarQube

Principais Vantagens da Análise Estática

- Redução de bugs e vulnerabilidades
- Melhor legibilidade e manutenção
- Aumento da produtividade
- Redução de custos de desenvolvimento
- Adoção fácil em CI/CD

Introdução ao PMD

O que é o PMD?

- Ferramenta open-source de análise estática
- Eficiente e gratuita
- Facilita a manutenção e melhora a qualidade do código

Linguagens suportadas

- Java
- Apex
- JavaScript
- (16 no total)

Detecção de:

- Código morto
- Más práticas (duplicações)
- Complexidade ciclomática
- Convenções de nomenclatura (má formatação)

Funcionamento do PMD

Escaneia o código-fonte

- Variáveis inutilizadas
- Blocos catch vazios
- Criação de objetos desnecessárias
- CPD incluído (copy-paste-detector)
- etc

Análise de árvore sintática (AST)

- AST (Abstract Syntax Tree)
- Comparação de padrões
- Nodes
- Violações

Mecanismo baseado no uso de regras

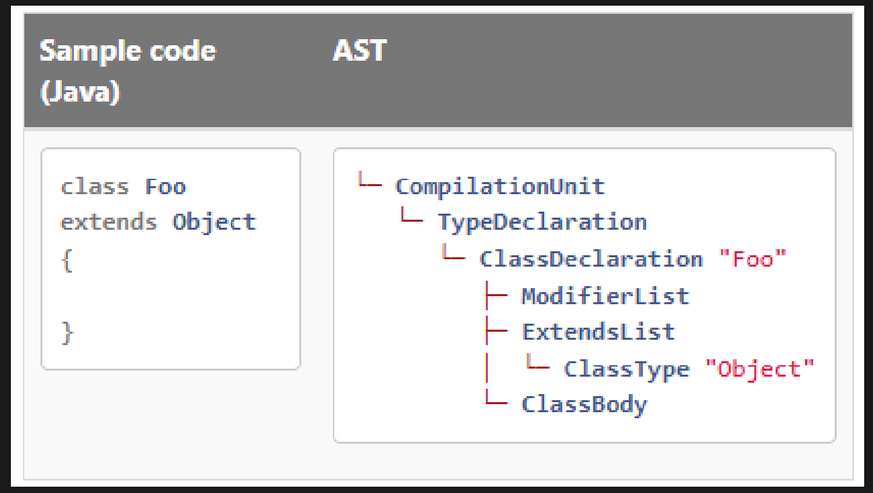
- Configuração por arquivos .xml (rulesets)
- +400 Build-in Rules
- O usuário pode expandir com rulesets e regras customizadas para seu projeto

Integração

- IDEs:
  - Eclipse, VSCode, IntelliJ...
- Pipelines:
  - Maven, Gradle

Análise de Árvore Sintática

- **AST (Abstract Syntax Tree)**
  - Estrutura em árvore que representa a estrutura sintática do código;
  - Criada pelo PMD antes de aplicar as regras;



- **Aplicação**
  - PMD compara padrões com a árvore AST;
  - Encontra nós (nodes) que satisfazem condições específicas;
  - Relata violações quando os padrões das regras são correspondidos.

Regras e Rulesets PMD

- **O que é um Ruleset?**
  - Arquivo XML que agrupa regras a serem executadas pelo PMD;
  - Permite customizar a análise além dos rulesets padrão;

- **O que é uma regra?**
  - Diretriz/Critério de avaliação;
  - Verificação automática de código-fonte;

- **Estrutura de uma regra**
  - Nome;
  - Descrição (o que ela verifica);
  - Prioridade (1 a 5);
  - Propriedades (opcional: configurações como limites de tamanho, nomes permitidos etc.);
  - Mensagem personalizada (descreve o problema para o usuário)

```
<rule ref="category/java/design.xml/NPathComplexity">
    <properties>
        <property name="reportLevel">
            <value>150</value>
        </property>
    </properties>
</rule>
```

Ruleset padrão (JAVA)

- **Categorias**
  - **Best Practices**
    - evitam armadilhas comuns e buscam manter código limpo;
    - Exemplo: AbstractClassWithoutAbstractMethod;
  - **Code Style**
    - consistência de formatação e estilo;
    - Exemplo: NoPackage;
  - **Design**
    - estrutura do código orientado a objetos (acoplamento excessivo, herança mal utilizada etc.);
    - Exemplo: AbstractClassWithoutAnyMethod;
  - **Documentation**
    - Garante que haja documentação mínima e bem formatada;
    - Exemplo: CommentSize;

Ruleset padrão (JAVA)

- **Categorias**
  - **Error Prone**
    - Detecta padrões que podem levar a bugs;
    - Exemplo: AvoidFieldNameMatchingMethodName;
  - **Multithreading**
    - Aponta problemas em código concorrente (sincronização incorreta ou uso perigoso de threads);
    - Exemplo: AvoidSynchronizedAtMethodLevel;
  - **Performance**
    - Identifica trechos de código com impacto negativo no desempenho;
    - Exemplo: AvoidInstantiatingObjectsInLoops;
  - **Security**
    - Alerta sobre práticas inseguras que podem levar a vulnerabilidades;
    - Exemplo: AvoidUsingHardCodedIP;

Como Configurar e Executar o PMD

- **Instalação via CLI (Command Line Interface)**
  - Extração do arquivo .zip (bin - Github)
  - Inclusão no PATH
    - ~/.zshrc ou ~/.bashrc - recomendado
  - Adição do Shell Completion
    - source <(pmd generate-completion) - recomendado
- **Instação via IDE (IntelliJ)**
  - Adição do plugin
  - Clique com botão direito na pasta do projeto
    - Run PMD
    - Escolha das regras

Comando básico CLI:

pmd check -f <format> -R <path> <source>

Execução com Maven:

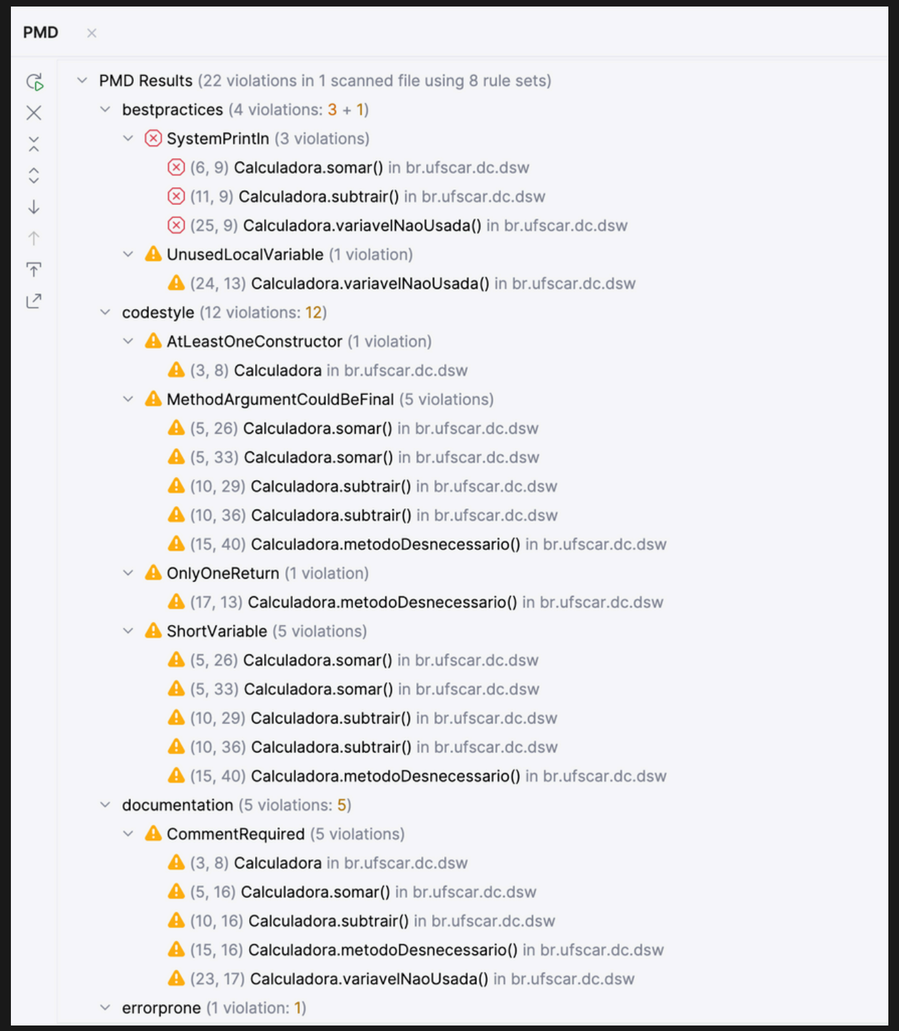
mvn compile pmd:pmd

Exemplo de Código para Análise

- Código Java com más práticas (método desnecessário, variável não usada)
- Fazer teste local para melhor experiência
  - Clone o repositório

```java
public class Calculadora {  no usages  new *

    public int somar(int a, int b) {  no usages  new *
        System.out.println("Somando...");
        return a + b;
    }

    public int subtrair(int a, int b) {  no usages  new *
        System.out.println("Subtraindo...");
        return a - b;
    }

    public int metodoDesnecessario(int x) {  no usages  new *
        if (true) {
            return x;
        } else {
            return 0;
        }
    }

    public void variavelNaoUsada() {  no usages  new *
        int inutil = 42;
        System.out.println("Olá");
    }
}
```

Análise do Código com PMD



Avoid unused local variables such as 'inutil'.
Detects when a local variable is declared and/or assigned, but not used. Variables whose name starts with `ignored` or `unused` are filtered out.
Full documentation

```
// Example:
public class Foo {
   public void doSomething() {
      int i = 5; // Unused
   }
}
```

Personalização de Rulesets e Regras

- **Criando um ruleset**
  - Arquivo de configuração .xml
    - Referências as regras
    - Exclusão de regras

```xml
<?xml version="1.0"?>

<ruleset name="Custom Rules"
    xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0
https://pmd.sourceforge.io/ruleset_2_0_0.xsd">

    <description>
        My custom rules
    </description>


    <!-- Your rules will come here -->

</ruleset>
```

Personalização de Rulesets e Regras

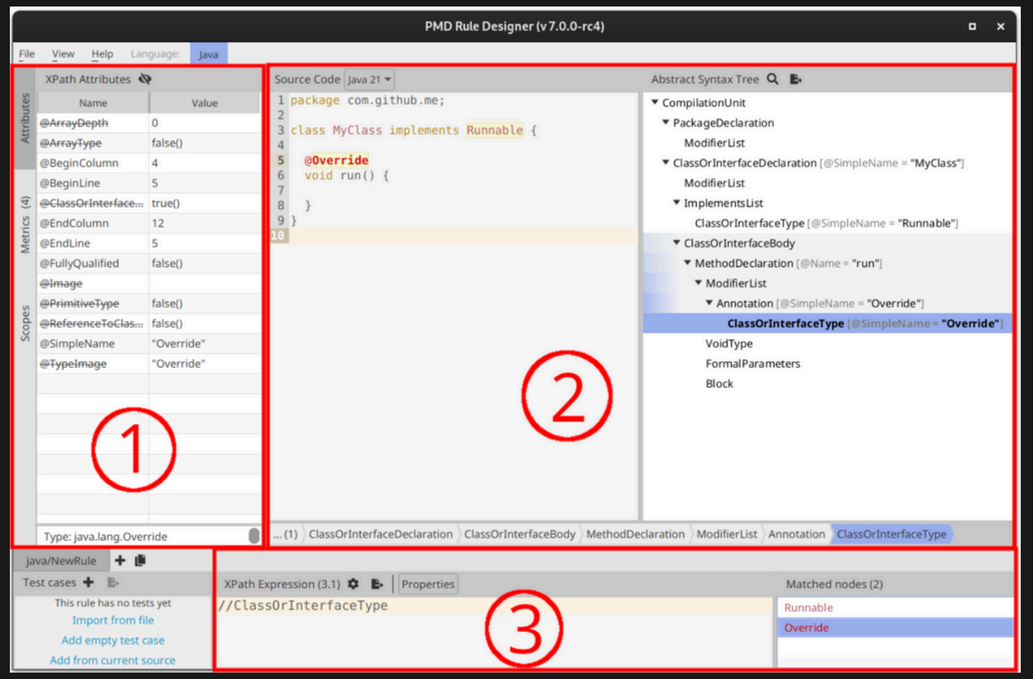
- **Personalizando uma regra**
  - Mesmo arquivo de configuração .xml (possível fazer via CLI)
    - Mensagem da regra
    - Prioridade da regra
    - Propriedades da regra
      - Cada regra tem suas próprias propriedades documentadas

```
<rule ref="category/java/errorprone.xml/EmptyCatchBlock"
      message="Empty catch blocks should be avoided" >
      <priority>5</priority>
</rule>
```

```
<rule ref="category/java/design.xml/NPathComplexity">
    <properties>
        <property name="reportLevel">
              <value>150</value>
        </property>
    </properties>
</rule>
```

Personalização de Rulesets e Regras

- **Criando uma regra**
  - Pode usar XPath Query ou Java Visitor
    - Recomendado XPath
      - Seletor de nós
      - Definida diretamente no .xml
  - Usa Rule Designer
    - $pmd designer




```
<rule name="DontCallBossShort"
      language="java"
      message="Boss wants to talk to you."
      class="net.sourceforge.pmd.lang.rule.xpath.XPathRule">
    <description>
        TODO
    </description>
    <priority>3</priority>
    <properties>
        <property name="xpath">
            <value>
<![CDATA[
//VariableId[@Name = "bill"][../../Type[@TypeImage="short"]]
]]>
            </value>
        </property>
    </properties>
</rule>
```

Integração do PMD Com o Processo de Desenvolvimento

- **Automatização com Maven**
  - Rodar o PMD automaticamente na fase de build do projeto
  - Gerar relatórios
    - Rulesets personalizados
  - Alterar pom.xml

```xml
<reporting>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-pmd-plugin</artifactId>
            <version>3.23.0</version>
            <configuration>
                <rulesets>
                    <ruleset>/rulesets/java/quickstart.xml</ruleset>
                    <ruleset>d:\rulesets\my-ruleset.xml</ruleset>
                    <ruleset>http://localhost/design.xml</ruleset>
                </rulesets>
            </configuration>
        </plugin>
    </plugins>
</reporting>
```

```xml
<project>
  ...
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-pmd-plugin</artifactId>
        <version>3.23.0</version> <!-- or use version from pluginManagement -->
        <configuration>
            <!-- failOnViolation is actually true by default, but can be disabled -->
            <failOnViolation>true</failOnViolation>
            <!-- printFailingErrors is pretty useful -->
            <printFailingErrors>true</printFailingErrors>
        </configuration>
        <executions>
          <execution>
            <goals>
              <goal>check</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

Exemplo de Integração PMD com Maven

- Fazer teste local para melhor experiência
  - mvn clean verify site
  - Clone o repositório

```xml
<build>
  <plugins>
    <!-- PMD Plugin Ativado -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-pmd-plugin</artifactId>
      <version>3.23.0</version>
      <configuration>
        <failOnViolation>true</failOnViolation>
        <printFailingErrors>true</printFailingErrors>
        <rulesets>
          <!-- Use o padrão interno, ou substitua por um ruleset customizado -->
          <ruleset>/rulesets/java/quickstart.xml</ruleset>
        </rulesets>
      </configuration>
      <executions>
        <execution>
          <phase>verify</phase>
          <goals>
            <goal>check</goal>
            <goal>cpd-check</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
```

```xml
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-pmd-plugin</artifactId>
      <version>3.23.0</version>
      <configuration>
        <rulesets>
          <ruleset>/rulesets/java/quickstart.xml</ruleset>
        </rulesets>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```

Integração do PMD com Maven

```
[INFO] <<< pmd:3.23.0:check (default) < :pmd @ TestePMD_ATA1 <<<
[INFO]
[INFO]
[INFO] --- pmd:3.23.0:check (default) @ TestePMD_ATA1 ---
[INFO] PMD version: 7.0.0
[INFO] PMD Failure: br.ufscar.dc.pooa.Calculadora:16 Rule:UnconditionalIfStatement Priority:3 Do not use if statements that are always true or always false.
[INFO] PMD Failure: br.ufscar.dc.pooa.Calculadora:24 Rule:UnusedLocalVariable Priority:3 Avoid unused local variables such as 'inutil'..
[INFO] PMD Failure: br.ufscar.dc.pooa.Main:3 Rule:UseUtilityClass Priority:3 This utility class has a non-private constructor.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.935 s
[INFO] Finished at: 2025-06-10T21:34:16-03:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-pmd-plugin:3.23.0:check (default) on project TestePMD_ATA1: You have 3 PMD violations. For more details see: /User
s/diogobellini/Desktop/UFSCar/POOA/TestePMD_ATA1/target/pmd.xml -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
```

Integração do PMD com Maven

Resultados do PMD

O seguinte documento contém os resultados do PMD 7.0.0.

Violations By Priority

Priority 3

br/ufscar/dc/pooa/Calculadora.java

Rule	Violação	Linha
UnconditionalIfStatement	Do not use if statements that are always true or always false	16–20
UnusedLocalVariable	Avoid unused local variables such as 'inutil'.	24

br/ufscar/dc/pooa/Main.java

Rule	Violação	Linha
UseUtilityClass	This utility class has a non-private constructor	3

Arquivos

br/ufscar/dc/pooa/Calculadora.java

Rule	Violação	Priority	Linha
UnconditionalIfStatement	Do not use if statements that are always true or always false	3	16–20
UnusedLocalVariable	Avoid unused local variables such as 'inutil'.	3	24

br/ufscar/dc/pooa/Main.java

Rule	Violação	Priority	Linha
UseUtilityClass	This utility class has a non-private constructor	3	3

Vantagens e Limitações

Vantagens	Fácil de configurar Customização de regras Código aberto e ativo
Limitações	Falsos positivos Não substitui testes unitários Não entende contexto de negócio Complementar com outras ferramentas (Ex: SpotBugs, SonarQube) Relativamente difícil criação de regras personalizadas

Obrigado!