

Mineração de Repositórios para Identificação de Code Smells

Boas Práticas de Programação - BPP 2025.2

Prof. Fernando Marques Filho

10 e 17 de Outubro de 2025

Universidade Federal do Rio Grande do Norte

Agenda

Introdução à Mineração de Repositórios

Ferramentas para MSR

MSR na Prática

Automação com CI/CD

Prática em Sala

Integração com BPP

Conclusão e Próximos Passos





Introdução à Mineração de Repositórios

O que é Mineração de Repositórios?





Definição

Mining Software Repositories (MSR) é a análise sistemática de dados históricos de repositórios para extrair padrões e insights sobre o desenvolvimento de software.

O que analisamos?

-  Commits e mudanças
-  Issues e bugs
-  Discussões
-  Histórico de mudanças

Para que serve?

-  Identificar code smells
-  Prever problemas
-  Avaliar qualidade
-  Tomar decisões

Análise Estática vs Mineração de Repositórios

</> Análise Estática Snapshot atual

- Examina código atual
- Problemas estruturais
- Code smells estáticos
- Violações de padrões

🕒 Mineração Evolução temporal

- Analisa histórico
- Padrões de mudança
- Smells temporais
- Correlações

Combinamos ambas para visão completa!

Por que Minerar Repositórios?

1. Identificar Hotspots

- Arquivos que mudam frequentemente
- Alta complexidade + muitas mudanças = risco

2. Detectar Code Smells Temporais

- Shotgun Surgery
- Divergent Change
- God Class evolutivo

3. Priorizar Refatorações

- Focar onde mais importa

4. Entender Conhecimento

- Especialistas por área
- Riscos de concentração

Ferramentas para MSR

Ferramentas Gratuitas - Visão Geral

Ferramenta	Linguagens	Foco
SonarCloud	30+	Análise completa
CodeClimate	Python, JS, Ruby	Manutenibilidade
PyDriller	Python	Mineração Git
CodeScene	Múltiplas	Análise comportamental

Importante

Todas gratuitas para projetos acadêmicos e open source!





Características:

- <https://sonarcloud.io>
- Gratuito para projetos públicos
- Integração nativa com GitHub
- 30+ linguagens suportadas

Métricas Principais

Reliability | Security | Maintainability | Coverage

O que detecta:

-  Bugs
-  Vulnerabilidades
-  Code Smells
-  Duplicação

O que é?

Biblioteca Python para análise automatizada de repositórios Git.

Instalação:

```
1 pip install pydriller
```

Recursos:

- Análise de commits
- Estatísticas de arquivos
- Métricas de desenvolvedores
- Complexidade temporal

Vantagens:

- Fácil de usar
- Flexível
- Integração Python

PyDriller - Exemplo Prático

```
1 from pydriller import Repository
2 from collections import Counter
3
4 # Analisar arquivos mais modificados
5 file_changes = Counter()
6
7 for commit in Repository(
8     "https://github.com/user/repo",
9     since="2024-01-01"
10 ).traverse_commits():
11     for file in commit.modified_files:
12         file_changes[file.filename] += 1
13
14 # Top 10 arquivos mais modificados
15 for file, count in file_changes.most_common(10):
16     print(f"{file}: {count} mudancas")
```

O que é?

- Análise comportamental de código
- Combina código + histórico
- Visualizações avançadas
- Free tier (3 repositórios)

Recursos:

- Hotspots
- Code Health
- Knowledge Map
- Change Coupling

Diferencial

Identifica problemas que análise estática não vê!

MSR na Prática

Identificando Hotspots

```
1 from pydriller import Repository
2
3 # Configurar analise
4 repo_url = "https://github.com/user/repo"
5 hotspots = {}
6
7 for commit in Repository(repo_url).traverse_commits():
8     for file in commit.modified_files:
9         filename = file.filename
10        if filename not in hotspots:
11            hotspots[filename] = 0
12        hotspots[filename] += 1
13
14 # Ordenar por frecuencia
15 sorted_hotspots = sorted(hotspots.items(),
16                           key=lambda x: x[1],
17                           reverse=True)
```

⚠ Hotspots Problemáticos

- **God Class:** Muitas responsabilidades
- **Shotgun Surgery:** Mudanças espalhadas
- **Feature Envy:** Comportamento em classe errada
- **Instabilidade:** Muitas mudanças = muitos bugs

💡 Priorização

Alto churn + Alta complexidade = **Prioridade máxima!**

Analizando Desenvolvedores

```
1 from collections import Counter
2 from pydriller import Repository
3
4 # Mapear conhecimento da equipe
5 dev_expertise = {}
6
7 for commit in Repository(".").traverse_commits():
8     author = commit.author.name
9     for file in commit.modified_files:
10         if file.filename.endswith('.py'):
11             if author not in dev_expertise:
12                 dev_expertise[author] = set()
13                 dev_expertise[author].add(file.filename)
14
15 # Exibir especialistas por arquivo
16 for dev, files in dev_expertise.items():
17     print(f"{dev}: {len(files)} arquivos")
```


Problema	Métrica	Ferramenta	Limite
Long Method	LOC	SonarCloud	> 50
God Class	LOC + métodos	CodeClimate	> 500
Duplicação	% duplicada	SonarCloud	> 3%
Complexidade	CC	Radon	> 10
Shotgun Surgery	Frequência	PyDriller	Alta



Dica Acadêmica

Use como guia, não como regra absoluta. Contexto importa!

Automação com CI/CD

GitHub Actions - Análise Automatizada

```
1 name: Code Quality Analysis
2 on: [push, pull_request]
3
4 jobs:
5   quality:
6     runs-on: ubuntu-latest
7     steps:
8       - uses: actions/checkout@v3
9       - uses: actions/setup-python@v4
10         with:
11           python-version: '3.x'
12
13       - name: Install dependencies
14         run: pip install pydriller pylint radon
15
16       - name: Run analysis
17         run: |
18           python analysis_script.py
19           pylint **/*.py
```

Benefícios da Automação

Para Desenvolvedores:

- ⚡ Feedback imediato
- 🛡️ Prevenção de problemas
- 📈 Acompanhamento contínuo
- 🎓 Aprendizado contínuo

Para o Projeto:

- ★ Qualidade consistente
- ⌚ Redução de revisões
- 👥 Padrão uniforme
- 🕒 Histórico de métricas

Prática em Sala

Exercício 1: Configuração Rápida (20 min)

Configurar Ferramentas

1. Acesse <https://sonarcloud.io>
2. Login com GitHub
3. Analyze new project
4. Selecione repositório da disciplina
5. Execute análise inicial
6. Anote 3 code smells principais

Importante

Repositório deve ser público para SonarCloud gratuito!

Exercício 2: Análise com PyDriller (25 min)

🔍 Identificar Hotspots

1. Clone seu repositório BPP
2. Instale PyDriller: `pip install pydriller`
3. Execute script de análise
4. Identifique top 5 arquivos mais modificados
5. Documente no README.md

📄 Script Base

```
from pydriller import Repository
for commit in Repository(".").traverse_commits():
    for file in commit.modified_files:
        print(f"file.filename - commit.author")
```

Exercício 3: Análise Comparativa (Para Casa)

Comparação entre Projetos

Escolha 2 repositórios populares e compare:

Qualidade:

- Technical debt
- Code smells
- Cobertura de testes
- Duplicação

Evolução:

- Hotspots
- Frequência de commits
- Tamanho das mudanças
- Distribuição de contribuições

Integração com BPP

CrITÉRIOS de Avaliação - U3

- Code Smells Identificados (20%)
- Refatorações Aplicadas (20%)
- Qualidade do Código (30%)
- Documentação da Análise (30%)

✔ Checklist Obrigatório

- SonarCloud configurado
- Hotspots identificados
- Code smells documentados
- Refatorações aplicadas
- Screenshots das métricas

README.md - Seção "Análise de Qualidade"

1. Ferramentas Utilizadas
2. Code Smells Identificados
3. Hotspots e Métricas
4. Refatorações Aplicadas
5. Evolução das Métricas

Dica Extra

Evolução positiva será valorizada! Mostre melhorias ao longo do tempo.

Conclusão e Próximos Passos

Principais Conceitos

1. MSR complementa análise estática
2. Hotspots indicam problemas reais
3. Ferramentas gratuitas são poderosas
4. Automação é essencial para qualidade
5. Métricas guiam decisões de refatoração

Ferramentas-Chave:

- SonarCloud
- PyDriller
- GitHub Actions

Próximos Tópicos:

- Técnicas de Refatoração
- Padrões de Projeto
- Testes Automatizados

17 de Outubro - Preparação

- Traga análise do seu projeto
- Liste code smells identificados
- Prepare perguntas específicas
- Tenha IDE cloud configurada

IDEs na Nuvem Recomendadas

- GitHub Codespaces
- GitPod
- Replit

Ferramentas:

- SonarCloud: sonarcloud.io
- PyDriller:
pydriller.readthedocs.io
- CodeScene: codescene.com

Documentação:

- Repositório BPP: github.com/fmarquesfilho/bpp-2025-2


Comunidade:

- Discord: discord.gg/bbMFJBQRT8

Referências:

- Awesome MSR
- Catálogo de Code Smells

Dúvidas?

 [fmarquesfilho/bpp-2025-2](https://github.com/fmarquesfilho/bpp-2025-2)

 discord.gg/bbMFJBQRT8

Bom trabalho!