Análise de Código: precisamos falar disso!

Guilherme Lacerda

Quem sou eu?

Guilherme Lacerda

@guilhermeslac | www.guilhermelacerda.net

umbler

WILDTECH

UNISINOS

“**Grande parte** do **dinheiro gasto** com desenvolvimento de **software** é **usado** para **entender códigos existentes**”

Kent Beck

Implementation Patterns, *Addison-Wesley Professional, 2008*

“Profissionais passam 40% da sua jornada semanal lidando com problemas de manutenção, como depuração e refatoração, além de correção de **‘código mal escrito’**. Segundo a pesquisa, o impacto disso equivale a quase US$ 85 bilhões em custo de oportunidade perdido anualmente em todo o mundo, de acordo com os cálculos sobre o salário médio do desenvolvedor por país”
The Developer Coefficient (Stripe, 2018)

Para refletir

Quanto tempo você leva para “aprender” sobre
o repositório de código que você trabalha?

E se você trocar de empresa, em quanto tempo você consegue
efetivamente “colocar a mão na massa”?

O que é análise de código?

Essência da Análise: Legibilidade e Compreensão

- Estruturas **pequenas**
- Nomes **significativos**
- **Formatação** e uso de padrões *(code conventions)*
- **Organização** das estruturas e algoritmos
- Aplicação dos **princípios do paradigma**
- **Testes** automatizados

Pilares da Análise

- Coesão
- Acoplamento
- Tamanho
- Complexidade

Por que analisar código é importante?

- **Ampliar** nossa **capacidade cognitiva** de programação
- **Conhecer** outros **paradigmas, padrões e linguagens** (e **problemas** também!)
- **Ampliar** nossas **habilidades**

Quais habilidades eu preciso desenvolver?

- Conhecer **heurísticas de análise para as estruturas**
  - *Módulos/Pacotes, Classes, Métodos*
- Compreender aspectos de **qualidade de software**
  - *Atributos externos e internos*
- Aplicar **métricas** de análise, estratégias de **visualização e ferramentas de apoio**
  - *Compreensão de software, mineração de repositórios*
- **Estratégias**
  - *Análise Estática, Análise Dinâmica, Análise Temporal, Análise Comportamental*

Combinando Estratégias

S1
Análise Estática
Análise Dinâmica

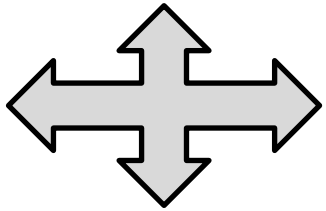
S2
Análise Estática
Análise Dinâmica

S3
Análise Estática
Análise Dinâmica

mainline

tempo

Análise Histórica



Análise Comportamental

Um Kata para Análise de Código

- Defina um **objetivo para análise**

- Rode alguma(s)**ferramenta(s) de análise** para encontrar o ponto que **você deseja**
  - *Níveis de granularidade*
  - *Uso de testes automatizados*
  - *Padrões adotados*

- Você pode começar de **“dentro para fora”**
  - *Analise as funções/métodos, suas estruturas e design*
  - *Considere os pilares*
  - *Suba a granularidade, quando necessário*

Qual o melhor momento para fazer análises?

- **Sempre!!**
  - *E de forma antecipada e quando possível*
- **Individualmente**, antes de fazer commits
  - *Inspeção na fonte (Poka-yoke, do Lean)*
  - *Apoiado por ferramentas (Jidôka, do Lean)*
- **Em par**, para discutir situações específicas
- Em sessões de **Code Review**

“As **ferramentas de qualidade** podem **ajudar a evitar** com que **problemas aconteçam**, simplesmente pela adoção dessas **ferramentas antes das confirmações** (commits), **evitando** ou **limitando a introdução** de **novos problemas no código**...”

Tufano et al

When and Why Your Code Starts to Smell Bad, *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Florence, 2015*

Ferramentas

Better Code Hub

DR-Tools

CODESCENE

SourceLevel

scitools
Understand

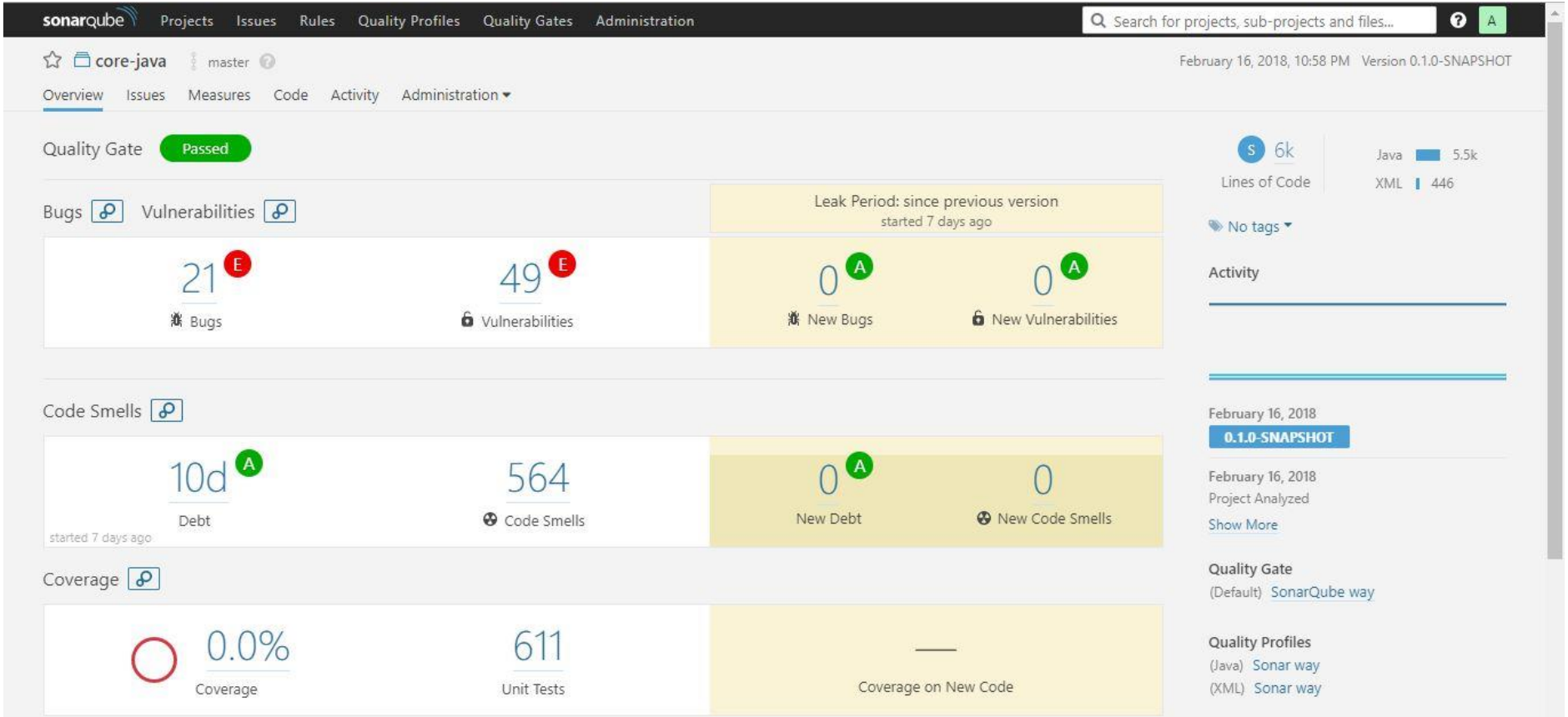
CODEMR

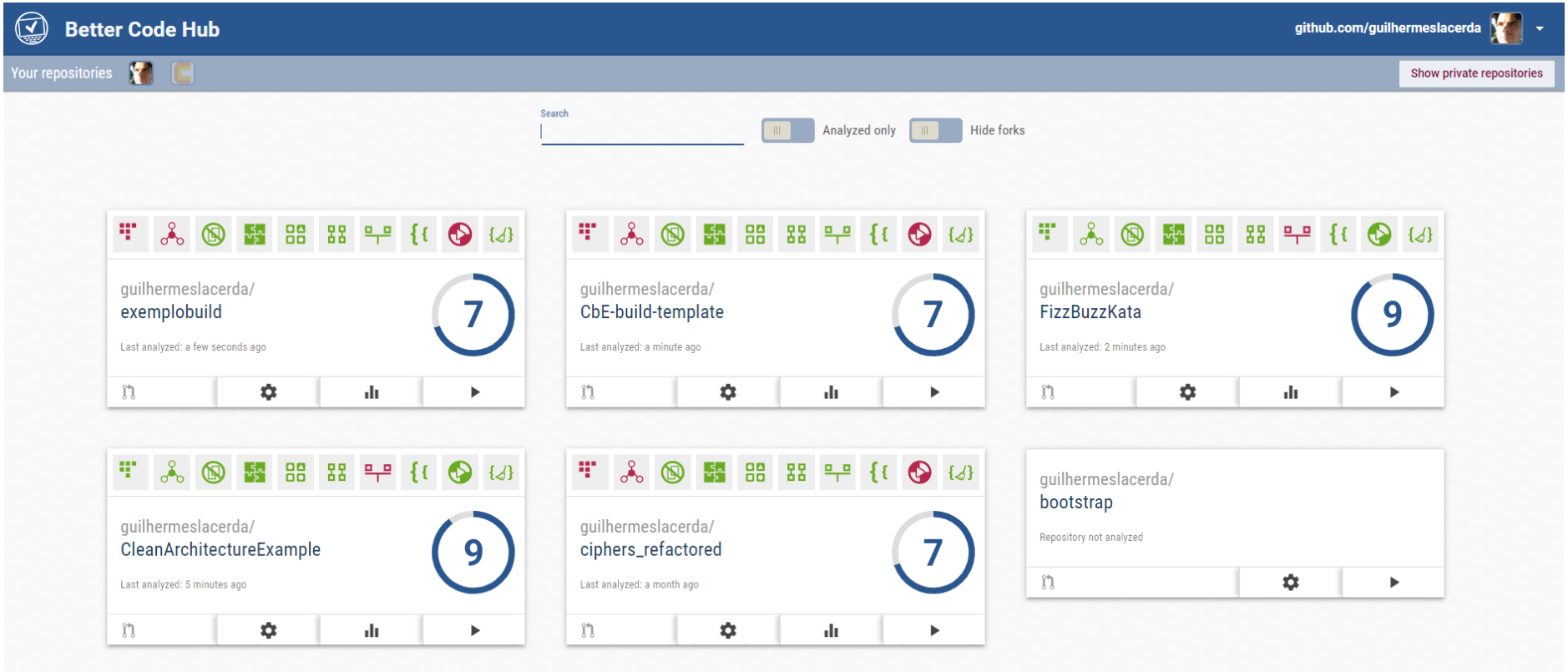
ArchUnit

SOKRATES

sonarqube

JUnit


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects, sub-projects and files...
A
core-java
master
February 16, 2018, 10:58 PM Version 0.1.0-SNAPSHOT
Overview
Issues
Measures
Code
Activity
Administration
Quality Gate
Passed
Bugs
Vulnerabilities
Leak Period: since previous version
started 7 days ago
21
E
Bugs
49
E
Vulnerabilities
0
A
New Bugs
0
A
New Vulnerabilities
Code Smells
10d
A
Debt
started 7 days ago
564
Code Smells
0
A
New Debt
0
New Code Smells
Coverage
0.0%
Coverage
611
Unit Tests
—
Coverage on New Code
6k
Lines of Code
Java 5.5k
XML 446
No tags
Activity
February 16, 2018
0.1.0-SNAPSHOT
February 16, 2018
Project Analyzed
Show More
Quality Gate
(Default) SonarQube way
Quality Profiles
(Java) Sonar way
(XML) Sonar way


Better Code Hub
github.com/guilhermeslacerda
Your repositories
Show private repositories
Search
Analyzed only
Hide forks
guilhermeslacerda/
exemplobuild
7
Last analyzed: a few seconds ago
guilhermeslacerda/
CbE-build-template
7
Last analyzed: a minute ago
guilhermeslacerda/
FizzBuzzKata
9
Last analyzed: 2 minutes ago
guilhermeslacerda/
CleanArchitectureExample
9
Last analyzed: 5 minutes ago
guilhermeslacerda/
ciphers_refactored
7
Last analyzed: a month ago
guilhermeslacerda/
bootstrap
Repository not analyzed

```
C:\Program Files\cmder
λ drtools-metric D:\JavaApps\Doutorado\repos\findbugs-3.0.1\src\ -sat --console --top 5
```

METRIC	1stQ	3rdQ	Avg	Median	Min	Max	Max-Min	StdDev	U-Fnc	Threshold
SLOC	21,00	129,00	118,30	56,00	3,00	3001,00	2998,00	195,00	291,00	500,00
NOM	3,00	11,00	10,13	5,00	0,00	202,00	202,00	16,38	23,00	14,00
NPM	2,00	9,00	8,64	4,00	0,00	198,00	198,00	14,71	19,50	40,00
WMC	4,00	27,00	25,75	11,00	1,00	629,00	628,00	46,23	61,50	100,00
DEP	2,00	13,00	9,75	5,00	0,00	103,00	103,00	12,38	29,50	20,00
I-DEP	0,00	6,00	4,48	2,00	0,00	69,00	69,00	6,78	15,00	15,00
FAN-IN	0,00	4,00	5,62	1,00	0,00	218,00	218,00	16,51	10,00	10,00
FAN-OUT	2,00	10,00	7,85	5,00	0,00	75,00	75,00	8,85	22,00	15,00
NOA	0,00	6,00	4,68	2,00	0,00	76,00	76,00	7,41	15,00	8,00
LCOM3	0,00	0,92	0,56	0,75	0,00	2,00	2,00	0,44	2,30	1,00

Legend:
1stQ=First Quartile | 3rdQ=Third Quartile | Avg=Average | Median=Median | Min=Min value | Max=Max value
Max-Min=Amplitude | StdDev=Standard Deviation | U-Fnc=Upper Fence | Threshold=Metric Threshold

TYPES	SLOC	NOM	NPM	WMC	DEP	I-DEP	FAN-IN	FAN-OUT	NOA	LCOM3
edu.umd.cs.findbugs.OpcodeStack	3001	157	104	629	58	22	45	46	76	0,76
edu.umd.cs.findbugs.BugInstance	1673	202	192	420	67	28	218	48	28	0,93
edu.umd.cs.findbugs.detect.FindNullDeref	1375	40	20	228	103	69	0	75	17	0,79
edu.umd.cs.findbugs.detect.DumbMethods	1308	30	21	575	49	29	0	42	33	0,47
edu.umd.cs.findbugs.PluginLoader	1296	53	20	160	62	21	3	36	24	0,78

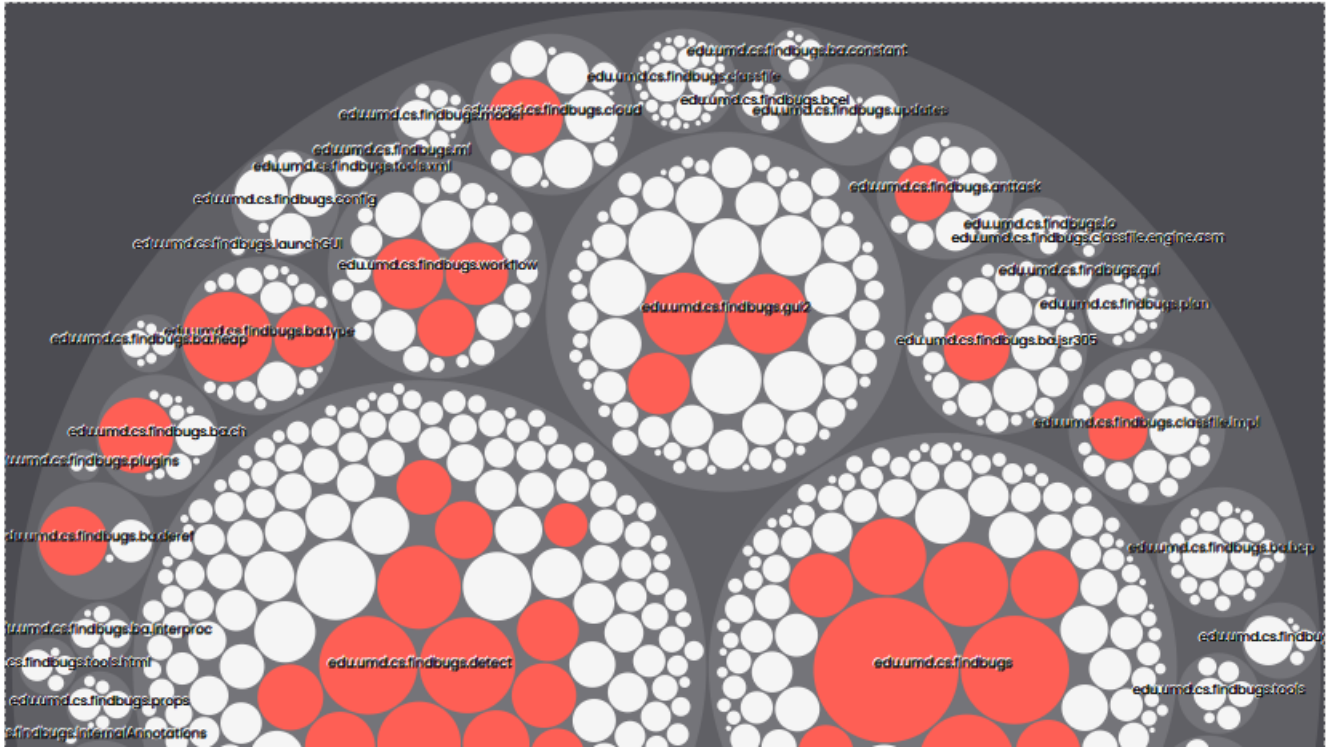
Code Resonance

Bubble Size (SLOC - Lines of Code) and Bubble Color (The most complex classes are red bubbles, with high WMC)

Project: Findbugs 3.0.1

Back

☐ Disable name of namespaces/types



Algumas Dicas

- Regra dos **30 segundos**
- Regra do **Escoteiro**/refatoração **oportunista**
- **Metáfora** do **jornal**
- **Olhe código de outros** (principalmente projetos open-source)
- **Pegou** um código **da Web**? **Ajuste-o**
  - *Antes de vincular ao seu repositório reestruture ele ao padrão do time*
- **Conheça** sua **ferramentas**
- Use **automação** em **diferentes níveis**
- **Defina políticas** de qualidade
  - *Quality Gates, Continuous Code Quality*

Algumas Dicas

- **Ao analisar** o código, **marque** pontos **para discussão** com o time
  - *Análise de Código X Revisão de Código*
- **Estudem** e **pratiquem**!
- Monte o **plano de metas** com o time
- Criem uma **rotina** com o time para discutir **problemas, práticas** e **ferramentas**
- **Experimentem** (novas LPs, IDEs, ambientes) através de **Dojos**
- **Participem** das **comunidades** e **eventos**
- **Entendam** que é uma **jornada a seguir**

Questões??

Análise de Código: precisamos disso!

Guilherme Lacerda
@guilhermeslac | www.guilhermelacerda.net