# Análise de Métricas de Código Fonte: Além das Métricas de *Design* de Código

Lucas Kanashiro Duarte

Universidade de Brasília

*kanashiro.duarte@gmail.com*

25 de Novembro de 2014

# Agenda

1. Introdução
   - Problema
   - Objetivos
   - Metodologia
2. Métricas
   - Métricas de Software
   - Métricas de Código fonte
   - Métricas de Vulnerabilidade
3. Estudo de Caso
   - Hipóteses
   - Metodologia
   - Teste de Hipóteses
   - Análise Qualitativa
4. Considerações Preliminares
   - Resultados obtidos
   - Atividades futuras

- Surgimento de novas classes de métricas de código fonte
- Como interpretar???
- Como monitorar???

Como deve-se interpretar e acompanhar outras métricas de código fonte que não as de *design*?

- Métricas de vulnerabilidade
- Métricas de anotação

- Revisão Bibliográfica
- Estudo de caso
- Modelar problema utilizando *Machine Learning*

# Como definir uma métrica?!

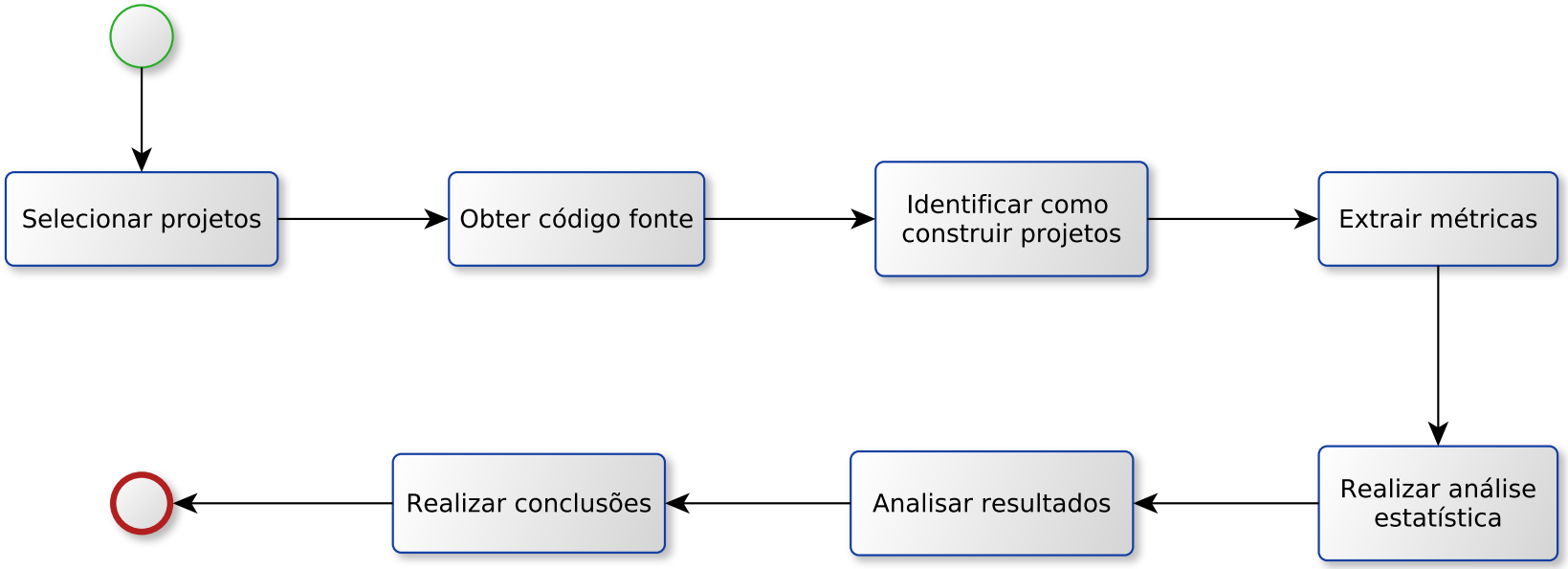
- Descrição
- Escala
- Como obter
- Como determinar
- Cálculo
- **INTERPRETAÇÃO**

- Métricas extraídas a partir do código fonte
- Utilização de ferramentas especializadas
- Métricas de *design* são as mais conhecidas e utilizadas

- Foco em segurança no últimos tempos
- Aplicações seguras != Aplicações de segurança
- Importante entender métricas de *design* para encontrar falhas
- Definição de cenários de vulnerabilidade a partir de lista de CWEs

Erros são inevitáveis, mas deve existir uma forma de medi-los e controla-los (Chess e West, 2007)

- *H1*: As métricas de vulnerabilidade de código fonte podem ser observadas de maneira similar as métricas de design de código fonte.
- *H2*: A média dos valores das métricas de vulnerabilidade de código fonte, geralmente, não é representativa para o acompanhamento das mesmas.
- *H3*: Os valores das métricas de vulnerabilidade de código fonte se comportam como distribuições estatísticas de cauda longa, e não distribuições estatísticas normalizáveis.


Selecionar projetos
Obter código fonte
Identificar como construir projetos
Extrair métricas
Realizar análise estatística
Analisar resultados
Realizar conclusões

Ferramentas utilizadas na análise do projeto *Linux Kernel*:

- Clang
- *Scripts* para análise estatística
- Analizo (necessidade de melhoria)

A hipótese *H1* foi negada, a *H2* foi confirmada e a *H3* não foi testada.

| Métrica | Mínimo | 1% | 5% | 10% | 25% | 50% | 75% | 90% | 95% | 99% | Máximo |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **AN** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **ASOM** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **AUV** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **BD** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **BF** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **DBZ** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **DF** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **DNP** | 0 | 0 | 0 | 0 | 0 | 1 | 1,75 | 2 | 2,75 | 11,1499 | 17 |
| **DUPV** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,3499 | 2 |
| **FGBO** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **MLK** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **OBAA** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **OSF** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **PITFC** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **ROGU** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **RSVA** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **SAIGV** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **UA** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **UAF** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **UAV** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | Mín | 1% | 5% | 10% | 25% | 50% | 75% | 90% | 95% | 99% | Máx |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Linux | 1,0 | 1,0 | 1,0 | 1,0 | 1,3 | 2,3 | 3,6 | 5,3 | 7,0 | 14,0 | 173,0 |
| FreeBSD | 0,0 | 0,0 | 0,0 | 0,0 | 1,0 | 2,0 | 4,5 | 8,0 | 11,1 | 22,8 | 830,0 |
| Android | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 4 | 6 | 13 | 110 |
| Bash | 1 | 1 | 1 | 1 | 2 | 3 | 6 | 10 | 18 | 30 | 214 |
| Chromium OS | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 6 | 8 | 15 | 163 |
| GCC | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 4 | 6 | 13 | 214 |
| Gimp | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 5 | 7 | 12 | 226 |
| Git | 1 | 1 | 1 | 2 | 3 | 4 | 6 | 9 | 10 | 22 | 46 |
| Gnome | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 5 | 6 | 12 | 527 |
| HTTP Server | 1 | 1 | 1 | 1 | 2 | 4 | 6 | 9 | 12 | 18 | 26 |
| KVM | 1 | 1 | 1 | 1 | 2 | 3 | 6 | 10 | 13 | 20 | 20 |
| MPlayer | 1 | 1 | 1 | 1 | 2 | 4 | 6 | 9 | 12 | 21 | 91 |
| OpenLDAP | 1 | 1 | 1 | 1 | 2 | 5 | 9 | 17 | 23 | 36 | 89 |
| PHP | 1 | 1 | 1 | 1 | 1 | 2 | 5 | 10 | 16 | 37 | 640 |
| Postgresql | 1 | 1 | 1 | 1 | 1 | 3 | 6 | 10 | 15 | 32 | 106 |
| Python | 1 | 1 | 1 | 1 | 1 | 3 | 5 | 8 | 11 | 20 | 424 |
| Subversion | 1 | 1 | 1 | 1 | 1 | 2 | 4 | 7 | 9 | 17 | 68 |
| VLC | 1 | 1 | 1 | 1 | 1 | 2 | 4 | 7 | 8 | 14 | 36 |

Figura: Tabela de percentis da métrica ACCM (MEIRELLES, 2013)

- Apenas métricas *Dereference of Null Pointer* e *Dereference of Undefined Pointer Value* não nulas
- Se concentram nas faixas mais altas de percentis
- Valores das métricas não variam muito entre os módulos

- Selecionou-se mais 10 projetos de software livre
- Utilizou-se basicamente a mesma metodologia
- Alterou-se o processo de análise estatística

| Projeto | Nº total de Módulos | Nº total de Módulos vulneráveis | % de Módulos vulneráveis |
|---|---|---|---|
| **Bash** | 253 | 36 | 14.2292 |
| **Blender** | 1472 | 106 | 7.2010 |
| **FFmpeg** | 1776 | 100 | 5.6306 |
| **Firefox** | 12404 | 328 | 2.6443 |
| **Gstreamer** | 264 | 22 | 8.3333 |
| **Inetutils** | 2576 | 27 | 1.0481 |
| **Linux Kernel** | 23056 | 43 | 0.1865 |
| **OpenSSH** | 248 | 14 | 5.6451 |
| **OpenSSL** | 974 | 26 | 2.6694 |
| **Python2.7** | 551 | 32 | 5.8076 |
| **Ruby-2.1** | 381 | 20 | 5.2493 |

Tabela: Análise de Módulos por Projeto

| Projeto | Nº total de Módulos | Nº total de Módulos vulneráveis | % de Módulos vulneráveis |
|---|---|---|---|
| **Bash** | 253 | 22 | 8.6956 |
| **Blender** | 1472 | 93 | 6.3179 |
| **FFmpeg** | 1776 | 37 | 2.0833 |
| **Firefox** | 12404 | 218 | 1.7574 |
| **Gstreamer** | 264 | 16 | 6.0606 |
| **Inetutils** | 2576 | 6 | 0.2329 |
| **Linux Kernel** | 23056 | 40 | 0.1734 |
| **OpenSSH** | 248 | 7 | 2.8225 |
| **OpenSSL** | 974 | 14 | 1.4373 |
| **Python2.7** | 551 | 18 | 3.2667 |
| **Ruby-2.1** | 381 | 10 | 2.6246 |

Tabela: Métrica DNP por projeto

- A partir da análise qualitativa se encontrou um lista de métricas de vulnerabilidade mais frequentes
  1. DNP (*Dereference of Null Pointer*)
  2. AN (*Argument with 'nonnull' attribute passed null*)
  3. ROGU (*Result of operation is garbage or undefined*)
  4. AUV (*Assigned value is garbage or undefined*)
- Análise qualitativa se apresentou mais produtiva do que a quantitativa

- Adicionar novos projetos de software livre
- Adicionar métricas de anotações
- Modelar a problemática de como interpretar e analisar as classes de métricas apresentadas usando uma abordagem de Aprendizado de Máquina

| Atividade | Data de Início | Data de Término |
|---|---|---|
| Selecionar novos projetos para análise de vulnerabilidades | 02/02 | 08/02 |
| Extrair e Analisar métricas de todos os projetos | 09/02 | 15/03 |
| Adicionar Métricas de Anotações | 16/03 | 29/03 |
| Selecionar projetos para análise de anotações | 30/03 | 05/04 |
| Extrair e Analisar métricas de anotação | 06/04 | 03/05 |
| Desenvolver modelo para métricas de vulnerabilidade | 04/05 | 31/05 |
| Desenvolver modelo para métricas de anotação | 01/06 | 21/06 |
| Entrega final do trabalho | 22/06 | 28/06 |

Tabela: Cronograma

# References

CHESS, Bryan; WEST, Jacob

Secure programming with static analysis

MEIRELLES, Paulo R. M.

Monitoramento de métricas de código-fonte em projetos de software livre

# Fim.