Universidade Federal da Bahia
Instituto de Matemática

Programa de Pós-graduação em Ciência da Computação

CARACTERIZAÇÃO DA QUALIDADE INTERNA DE FERRAMENTAS DE ANÁLISE ESTÁTICA DE CÓDIGO FONTE

Joenio Marques da Costa
joenio@joenio.me

QUALIFICAÇÃO DE MESTRADO

Salvador
1 de junho de 2016

Universidade Federal da Bahia
Instituto de Matemática

Joenio Marques da Costa
joenio@joenio.me

CARACTERIZAÇÃO DA QUALIDADE INTERNA DE FERRAMENTAS DE ANÁLISE ESTÁTICA DE CÓDIGO FONTE

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Instituto de Matemática da Universidade Federal da Bahia como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Christina von Flach G. Chavez
Co-orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Salvador
1 de junho de 2016

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

1.1 MOTIVAÇÃO

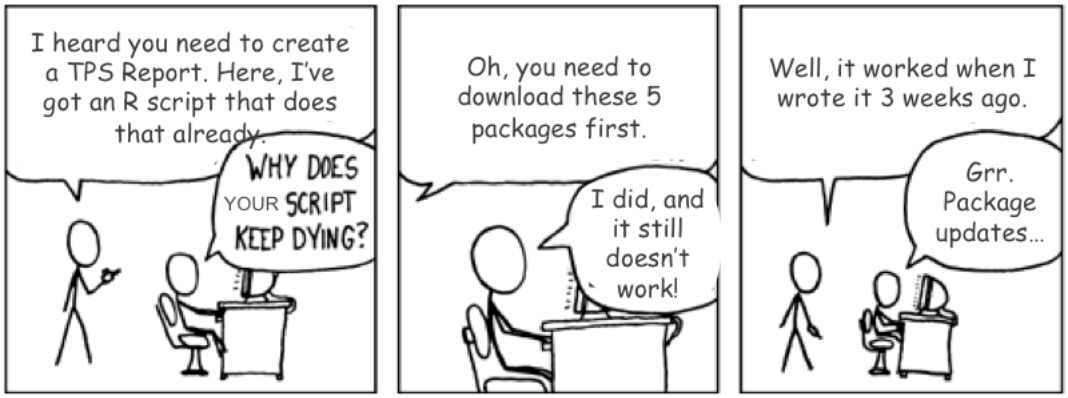

Figura 1.1 The reproducible research problem(VRIES, 2014)

Reprodutibilidade é a habilidade de replicar um experimento ou estudo em sua totalidade a fim de confirmar suas hipóteses e resultados, apesar de ser uma prática central do método científico ainda é um grande obstáculo em muitos estudos. Enquanto pesquisadores publicam artigos descrevendo e divulgando seus resultados, é raro que façam o mesmo com toda a produção gerada durante a pesquisa. A maioria dos componentes necessários para a reprodução dos resultados de uma pesquisa na engenharia de software – por exemplo, código-fonte e dados – usualmente permanecem não publicados.

Isto se configura como uma barreira para a reprodutibilidade, e consequentemente para a repetição, replicação e variação de estudos (FEITELSON, 2015) já que a disponibilidade de código-fonte é o mínimo necessário para que isto ocorra (PENG, 2011).

Dentro deste contexto, e considerando que muitos estudos ainda sofrem com dificuldades de repetição (KAZMAN, 2016), surge a preocupação de avaliar a qualidade dos softwares científicos a partir de métodos adequados, especialmente em relação a sua manutenabilidade e disponibilidade por serem problemas comuns enfrentados pelos pesquisadores (PRLIć; PROCTER, 2012).

1.2 CONTRIBUIÇÕES ESPERADAS

1. Valores de referência de métricas de código-fonte para ferramentas de análise estática.
2. Guia de sugestões de refatoração para ferramentas de análise estática.
3. ...

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

2.1 ENGENHARIA DE SOFTWARE

Sistemas de software são utilizados em praticamente todas as áreas do conhecimento humano e têm exercido um papel essencial em nossa sociedade (MAFRA; TRAVASSOS, 2006). A dependência crescente de serviços oferecidos por tais sistemas evidencia a necessidade de produzir software de qualidade, contornando os desafios relacionados a funcionalidades incompletas ou incorretas, custos acima do esperado ou prazos não cumpridos.

Diante desses desafios, surge a Engenharia de Software, uma disciplina centrada no desenvolvimento de sistemas de software através de uma abordagem sistemática, disciplinada, e quantificável para o desenvolvimento, operação e manutenção (SOCIETY, 2014).

Nas últimas décadas, o foco em estudos empíricos na área de Engenharia de Software tem crescido significantemente (STOL; FITZGERALD, 2015), resultando no uso crescente de métodos como surveys, estudos de caso, experimentos e revisões sistemáticas de literatura. Através destes estudos empíricos, pesquisadores transformam a Engenharia de Software em uma disciplina mais científica e controlável – a Engenharia de Software Experimental – provendo meios para avaliar e validar métodos, técnicas, linguagens e ferramentas.

Não raro, muitos destes estudos criam novos sistemas de software, tais sistemas costumam ser utilizados como meio para atingir os resultados da pesquisa ou, em alguns casos, são o próprio fim do estudo realizado. Neste trabalho, tais ferramentas de software são nosso objeto de pesquisa e serão chamados de "software científico" – Portillo-Rodríguez et al. (2012) utiliza o termo "research tool" para designar este mesmo tipo de software.

2.1.1 Software científico

Softwares científicos são ferramentas de software desenvolvidas no decorrer de pesquisas científicas como parte de um estudo, podem ser pequenos scripts, protótipos, ou mesmo produtos de software completos que demonstram ou refletem os resultados de uma pesquisa. Em Engenharia de Software este tipo de software desempenha um papel essencial e sua importância pode ser notada através do grande número de conferências com sessões específicas sobre publicação de ferramentas.

Kon et al. (2011) em um estudo sobre como pesquisas em Engenharia de Software podem se beneficiar do ecosistema de Software Livre faz uma análise de 10 edições do SBES[1] e conclui que apesar do aumento do interesse por parte dos pesquisadores em

[1]Simpósio Brasileiro de Engenharia de Software

disponibilizar o código-fonte de suas ferramentas isto ainda é minoria. O que confirma a preocupação de Krishnamurthi e Vitek (2015) em um estudo sobre repetibilidade de pesquisas científicas, onde chamam atenção para o papel central que os artefatos de software possuem em pesquisas de ciência da computação e questionam: "Onde está o software nas pesquisas sobre linguagem de programação?".

A partir daí podemos afirmar que softwares científicos são peça fundamental para que pesquisadores independentes possam reproduzir, validar ou expandir os resultados encontrados em estudos anteriores e assim aumentar o rigor e a qualidade científica de tais pesquisas (VITEK; KALIBERA, 2011).

2.1.2 Qualidade de software

Qualidade de software diz respeito à quão bem um software é projetado e o quanto este software está em conformidade com o projeto, embora existam inúmeras definições há um concenso de que existem duas dimensões básicas para medir a qualidade de um software, estas dimensões estão relacionadas à características de qualidade interna e de qualidade externa.

Segundo McConnell (2004), qualidade interna são aquelas características que preocupam o desenvolvedor, como: manutenabilidade, flexibilidade, portabilidade, reusabilidade, legibilidade, testabilidade e compreensão. São questões relacionadas ao código-fonte e em como o software foi construído, como design e boas práticas por exemplo.

Qualidade externa são aquelas características que afetam o usuário, como por exemplo: exatidão, usabilidade, eficiência, confiabilidade, integridade, adaptabilidade, acurácia e robustez. Estas características impactam extritamente no uso do software e não em como o software foi construído.

Segundo a ISO/IEC 25010 (ISO, 2011) essas características podem ser divididas em subcaracterísticas. A característica usabilidade por exemplo é dividida nas seguintes subcaracterísticas: capacidade de compreensão, capacidade de aprendizado, operabilidade, atratividade e conformidade. Esta característica está relacionada, por exemplo, a facilidade com que usuários aprendem e utilizam um sistema, e passa por questões como facilidade de instalação, aprendizado, e uso.

Sabe-se que, em algum nível, as caractarísticas de qualidade interna afetam as características de qualidade externa (MCCONNELL, 2004), softwares que não possuem boa manutenabilidade por exemplo afetam a habilidade de correção de defeitos, que por sua vez afetam as características de exatidão e confiabilidade.

Dito isso podemos perceber que é possível inferir características de qualidade externa de um software a partir de seus atributos de qualidade interna, o que pode ser realizado a partir da análise de suas métricas de código-fonte.

2.1.3 Métricas de código-fonte

Uma métrica, segundo a definição da ISO/IEC 25010 (ISO, 2011), é a composição de procedimentos para a definição de escalas e métodos para medidas, em engenharia de software estas métricas podem ser classificadas em três categorias: métricas de produto, métricas de processo e métricas de projeto.

Métricas de produto são aquelas que descrevem as características de artefatos do desenvolvimento, como documentos, diagramas, código-fonte e arquivos binários. Métricas de processo medem atributos relacionados ao ciclo de desenvolvimento do software. Métricas de projeto são aquelas que descrevem as características dos recursos disponíveis ao desenvolvimento.

Neste trabalho, nosso interesse estão nas métricas de produto, que podem ser classificadas entre internas ou externas, ou seja, aquelas que medem propriedades visíveis apenas aos desenvolvedores ou que medem propriedades visíveis aos usuários, respectivamente. Iremos extrair propriedades dos softwares científicos a fim de medir a sua qualidade interna através de um conjunto de métricas previamente definido, este conjunto tomará como base o trabalho realizado por Meirelles (2013) onde foi realizado um estudo associando qualidade de software à qualidade de código-fonte através da observação de métricas de código-fonte. E será realizado através de ferramentas de análise estática de código-fonte.

2.1.4 Ferramentas de análise estática de código-fonte

Ferramentas de análise estática de código-fonte são ferramentas capazes de realizar a leitura de código-fonte de um projeto de software de forma automatizada ou semi-automatizada e extrair daí informações sobre as entidades do software, como módulos, classes, funções, métodos, variáveis, seus relacionamentos, suas características e diversas outras informações possíveis de serem exraídas diretamente do código-fonte que seja útil ao engenheiro de software.

Estas informaçoes costumam ser aplicadas à tarefas comuns da engenharia de software, como por exemplo, recuperação arquitetural, localização de falhas, manutenção, refatoração, compreensão, análise de performance, visualização, entre outras.

Segundo Kirkov e Agre (2010) estas ferramentas possuem uma anatomia comum, composta de quatro componentes básicos - construção de modelos; algoritmos de análise e reconhecimento de padrões; base de conhecimento de padrões; e representação final.

A construção de modelos de um programa é o primeiro passo e é feito por um parser de código-fonte (BINKLEY, 2007). A base de conhecimento de padrões é usada para representar e armazenar informações sobre potenciais problemas encontrados no código-fonte. O objetivo do algoritmo de análise e reconhecumento de padroes é classificar as informações encontradas no modelo a partir da base de conhecumento de padroes. A representação final é um relatório ou outro tipo de visualização apresentada através de uma interface de usuário apropriada.

Esta representação final pode ser por exemplo um relatório contendo os valores de métricas calculadas para o software sendo analisado, e é esta caractarística das ferramentas de análise estática de código-fonte que será utilizada neste trabalho para recuperar métricas de produto de software que reflitam os atributos de qualidade dos software científicos estudados.

(feedback de Paulo: faltou arrematar e contextualizar com o seu trabalho)

2.2 CAPÍTULO SOBRE ESTATÍSTICA

(ver TCC de Ronaldo e Kanashiro)

CAPÍTULO 3

METODOLOGIA

Visando então avaliar a qualidade de "softwares científicos" será feito um levantamento de artigos com publicação de ferramentas do domínio *análise estática de código-fonte*. Este domínio foi selecionado com base na experiência do pesquisador nesta área, o que facilita encontrar fontes sobre ferramentas, bem como analisar e estudar tais softwares. O foco em um domínio se justifica pela necessidade de reduzir o escopo do estudo a fim de viabilizar o trabalho dentro do tempo previsto em um programa de mestrado.

Além dos software científicos iremos também incluir ferramentas de análise estática desenvolvidas na indústria, o objetivo é aumentar o número de ferramentas analisadas visto que existe uma suspeita de encontrar um número pequeno de softwares científicos com disponibilidade de código-fonte.

Após o levantamento e seleção das ferramentas será feita a caracterização inicial e posteriormente análise estática do seu código-fonte, onde teremos métricas para cada ferramenta e possivelmente valores referência para ferramentas de análise estática, estes valores de referência darão origem a recomendações de refatoração para ferramentas deste mesmo domínio de aplicação.

Por final traçaremos um paralelo entre caracteristicas de qualidade externa e valores de métricas de qualidade interna, especialmente em relação à portabilidade e usabilidade a fim de responder nossa questão de pesquisa e validar suas hipóteses.

3.1 QUESTÃO DE PESQUISA E HIPÓTESES

A questão de pesquisa **Q1** - *Quais características das ferramentas de análise estática de código fonte podem ser obtidas a partir da avaliação da sua qualidade interna?* será respondida através da validação das seguintes hipóteses:

H1: *Quanto mais antiga for a publicação maior a probabilidade do software científico não estar mais disponível nas fontes indicadas*

H2: *É possível calcular valores de referência de métricas de código-fonte para ferramentas de análise estática a partir de um conjunto de softwares científicos e da indústria*

H3: *Ferramentas da indústria possuem valores de métricas de código-fonte mais próximos aos valores de referência do que os softwares científicos*

H4: *É possível relacionar a subcaracterística de qualidade externa operabilidade à valores de métricas de qualidade interna para os softwares científicos*

A hipótese **H1** será validada com a avaliação dos artigos que publicam software científico com fontes para obtenção mas que não se encontram mais disponíveis.

A hipótese **H2** será validada (depende do capítulo 2.2) ...

A hipótese **H3** será validade a partir do cálculo das métricas de código-fonte das ferramentas da indústria e sua comparação aos valores de referência encontrados para as ferramentas de análise estática.

A hipótese **H4** será validada com a tentativa de instalação e o uso das funções mínimas de cada software científico avaliado, será feito a tentativa de instalar a ferramenta, instruções de instalação e uso serão pesquisados nos artigos relacionados bem como junto ao código-fonte do software. Espera-se que aqueles softwares com maior dificuldade de instalação e uso terão valores de métricas piores.

3.2 PLANEJAMENTO DO ESTUDO

3.2.1 Seleção das métricas

Meirelles (2013) realizou uma série de estudos onde associou características de qualidade de produto de software à características de qualidade de código-fonte, através de métricas de código-fonte, em um destes estudos utilizou-se métricas que medem aspectos relevantes à manutenibilidade do software, como a preocupação aqui também está na manutenibilidade das ferramentas iremos utilizar a mesma seleção de métricas utilizada por Meirelles (2013).

- **CBO** *Coupling Between Objects (Acoplamento entre objetos)*: mede o acoplamento entre objetos do software (CHIDAMBER; KEMERER, 1994).
- **LCOM4** *Lack of Cohesion in Methods (Ausência de coesão em métodos)*: mede o grau de falta de coesão em métodos (HITZ; MONTAZERI, 1995).
- **SC** *Structural Complexity (Complexidade estrutural)*: mede a complexidade do software (DARCY et al., 2005).
- **AMLOC** *Average Method LOC (Média do número de linhas de código por método)*: indica se o código está bem distribuído entre os métodos, quanto maior mais "pesados" são os métodos (??)
- **ACCM** *Average Cyclomatic Complexity per Method (Média de complexidade ciclomática por método)*: mede a complexidade do programa (MCCABE, 1976).
- **RFC** *Response For a Class (Resposta para uma classe)*: número de métodos dentre todos os métodos que podem ser invocados em resposta a uma mensagem enviada por um objeto de uma classe (SHARBLE; COHEN, 1993).
- **DIT** *Depth of Inheritance Tree (Profundidade da árvore de herança)*: mede o número de ancestrais de uma classe (SHIH et al., 1997).
- **NOC** *Number Of Children (Número de filhos)*: número total de filhos de uma classe (ROSENBERG; HYATT, 1997).

- **COF** *Coupling Factor (Fator de acoplamento)*: razão entre o número máximo possível de acoplamentos no sistema e o número atual de acoplamentos possíveis por herança (HARRISON; COUNSELL; NITHI, 1998).

Estas métricas serão coletadas para cada classe/módulo presente nas ferramentas de análise estática selecionadas e servirão de base para avaliar a qualidade das mesmas.

3.2.2 Seleção da ferramenta de análise estática de código-fonte

Para realizar a caracterização das ferramentas e extrair suas métricas será necessário uma ferramenta que permita automatizar este processo, neste trabalho utilizaremos a ferramenta de análise estática Analizo (TERCEIRO et al., 2010).

Analizo é um *toolkit* livre, multi-linguagem e extensível para análise de código-fonte, calcula uma grande quantidade de métricas, como CBO, LCOM4, RFC, LOC, entre outras, e suporta análise das linguagens de programação C, C++ e Java.

É uma ferramenta mantida constantemente, com desenvolvedores ativos, e atualizações frequentes, sua última versão 1.19.0 lançada em 18 de Fevereiro de 2016 será a versão utilizada neste estudo.

3.2.3 Seleção das fontes de ferramentas de análise estática

Para ser possível validar as hipóteses aqui levantadas é necessário realizar uma busca por ferramentas de análise estática desenvolvidas no contexto da academia e da indústria, para isso, será feito um planejamento detalhado para realizar a seleção de ferramentas em cada um destes contextos.

No contexto acadêmica a busca por ferramentas será feita através de artigos publicados em conferências que tenham histórico de publicação sobre ferramentas de análise estática de código fonte. Estes artigos serão analisados e aqueles com publicação de ferramenta de análise estática serão selecionados.

Na indústria, a busca por ferramentas será feita a partir de referências encontradas na internet, algumas organizações mantém listas de ferramentas para análise de código-fonte, a Wikipedia por exemplo, mantém uma lista de ferramentas, estas referências serão utilizadas como ponto de partida e cada ferramenta será analisada a fim de validar se são da indústria ou surgiram em contexto acadêmico.

Uma vez que as ferramentas tenham sido selecionadas iniciaremos a extração de seus atributos de qualidade interna a partir do cálculo de suas métricas com o Analizo.

3.3 COLETA DE DADOS

A partir das fontes selecionadas na etapa anterior serão realizadas duas atividades para identificar e mapear as ferramentas de análise estática com código-fonte disponível, uma atividade relacionada ao levantamento de ferramentas da academia, outra atividade relacionada ao levantamento de ferramentas da indústria.

3.3.1 Ferramentas da academia

A seleção de ferramentas será relizada através de uma revisão estruturada dos artigos selecionados a partir das seguintes conferências:

- ASE - Automated Software Engineering[1]
- CSMR[2] - Conference on Software Maintenance and Reengineering[3]
- SCAM - Source Code Analysis and Manipulation Working Conference[4]
- ICSME - International Conference on Software Maintenance and Evolution[5]

Chamamos de revisão estruturada um processo disciplinado para seleção de artigos a partir de critérios bem definidos de forma que seja possível a reprodução do estudo por parte de pesquisadores interessados. Alguns resultados preliminares podem ser consultados na Tabela 4.1 da Seção 4.1.

3.3.2 Ferramentas da indústria

A seleção de ferramentas da indústria será feita de forma não estruturada a partir de uma busca livre e manual em fontes encontradas na Internet sobre ferramentas de análise estática. Nos estudos já realizados as seguintes fontes foram selecionadas:

- Projeto SAMATE[6] - *Software Assurance Metrics and Tool Evaluation* disponível em NIST (2016) mantém uma lista de ferramentas de análise estática, detalhes sobre o projeto pode ser encontrado em Ribeiro (2015).
- O software Spin mantém em seu site uma lista de ferramentas comerciais e de pesquisa para análise estática de código-fonte para C em Spin (2016).
- O Instituto de Engenharia de Software do CERT mantém uma lista de ferramentas de análise estática em CERT (2016).
- O software Flawfinder oferece em seu site um link com referências para inúmeras ferramentas livres, proprietárias e gratuitas de ferramentas de análise estática e outros tipos de análise em Wheeler (2015).
- Uma outra fonte contendo uma relação expressiva de ferramentas é mantida na Wikipedia em Wikipedia (2016).

Estas fontes serão pesquisadas manualmente em busca de ferramentas de análise estática que tenham sido desenvolvidas no contexto da indústria, alguns resultados preliminares podem ser encontrados nas Tabelas 4.3 e 4.2.

[1] http://ase-conferences.org
[2] A conferência CSMR tornou-se SANER - Software Analysis, Evolution, and Reengineering a partir da edição 2015.
[3] http://ansymore.uantwerpen.be/csmr-wcre
[4] http://www.ieee-scam.org
[5] http://www.icsme.org
[6] http://samate.nist.gov

3.3.3 Caracterização dos artigos

Os artigos selecionados a partir da revisão estruturada serão avaliados a fim de caracterizar se se tratam de publicação de ferramenta de análise estática de código-fonte, esta revisão será realizada de forma semi-automatizada, o primeiro passo será automatizado a partir de um script[7] que busca os seguintes termos no conteúdo dos artigos:

```
"tool" OU "framework"; E
"download" OU "available"; E
"http" OU "ftp"; E
"static analysis" OU "parser".
```

O segundo passo, manual, é realizar uma leitura do artigo a fim de identificar se realmente trata-se de um artigo com publicação de *software científico*, uma vez que se confirme que o artigo publica um software, identifica-se se o software é uma ferramenta de análise estática, softwares que sejam mais abrangentes do que apenas análise estática mas que contenham esta função em seu conjunto também será considerados.

Uma vez identificado os artigos que publicam *softwares científicos* de análise estática, procuramos no próprio artigo referências de onde encontrar o software, neste momento algumas ações serão tomadas a partir da situação encontrada.

- Aqueles autores que afirmam que a ferramenta está disponível mas o artigo não cita referências de onde encontrar serão contactados por email solicitando informações de onde obter o código-fonte.
- Os artigos que indicam onde obter o cófigo-fonte mas o acesso ao local indicado não está disponível, ou está disponível mas o software não se encontra lá, os autores também serão contactados solicitando informações atualizadas de onde obter uma cópia do código-fonte da ferramenta
- Os demais artigos que indicam onde obter o cófigo-fonte e a referência está correta, iremos fazer downlod da última versão disponível do software

Uma vez que os autores contactados por email respondam com informações de onde obter o software iremos adicinar estes softwares na lista de softwares a serem analisados.

3.3.4 Caracterização das ferramentas

As ferramentas da indústria e acadêmicas serão analisadas com o Analizo para extração das métricas de código-fonte préviamente selecionadas, estas métricas serão relacionadas à caractarística e atributos de qualidade das ferramentas.

As métricas coletadas serão também utilizadas para identificar se existem valores de referência para métricas de ferramentas de análise estática, caso existam, estes valores de métricas serão utilizadas para calcular quão distante cada ferramenta analisada se encontra dos valores de referência, isto será feito com base no trabalho realizado Júnior (2015) onde o mesmo estudo foi feito para o sistema Android e seus aplicativos.

[7] http://github.com/joenio/dissertacao-ufba-2016/blob/master/revisao-estruturada/filter

3.4 EXEMPLO DE USO

Por fim, os valores de referência encontradas serão documentados e servirão de base para criar um guia de sugestões para refatoração de ferramentas de análise estática. Tomaremos como exemplo de uso a própria ferramenta Analizo, onde iremos calcular suas métricas e a partir de uma comparação com os valores de referência indicaremos refatorações que façam a ferramenta de aproximar dos valores de referência.

Estas indicações serão feitas utilizando o mesmo método de Almeida e Miranda (2010) onde foi feito um estudo mapeando boas práticas de programação em valores de métricas de código-fonte, estas boas práticas são baseadas nos trabalhos de (MARTIN; HAN, 2012) e (BECK, 2007) sobre *Clean Code*, onde sugerem práticas de desenvolvimento úteis para que um código tenham expressividade, simplicidade e flexibilidade. Neste trabalho os autores identificam melhorias de implementação através do uso de valores de métricas e oferecem aos desenvolvedores uma maneira de pensarem em melhorias para os seus códigos.

CAPÍTULO 4

ANÁLISE

4.1 RESULTADOS PRELIMINARES

A Tabela 4.1 apresenta um resumo do número de artigos em cada edição do SCAM e quantos artigos trazem publicação de ferramenta de análise estática com código fonte disponível.

As Tabelas 4.3 e 4.2 apresentam ferramentas do NIST após avaliação inicial sobre disponibilidade do código-fonte. Das 54 ferramentas apenas 19 tinham código fonte disponível.

Assim, temos um total de 19 ferramentas da indústria com código-fonte disponível e ??? da academia com código fonte disponível, é preciso avaliar em qual linguagem de programação foi escrita cada ferramentas pois só iremos analisar aquelas em C, C++ ou Java que são suportadas pelo Analizo.

A ferramenta utilizada para identificar a linguagem de programação em que estes softwares foram escritos foi a sloccount, uma ferramenta livre para contagem de linhas de código fonte, onde se calcula em quais linguagens de programação um software foi escrito.

Após analise ficamos com um total de 25 ferramentas, 15 da indústria e 10 da academia, a Tabela 4.4 traz um resumo de todos as ferramentas analisadas.

Tabela 4.1 Total de artigos analisados por edições do SCAM

Edição	Total de artigos	Script filter	Artigos com ferramenta
SCAM 2001	23	6	-
SCAM 2002	18	6	-
SCAM 2003	21	7	-
SCAM 2004	17	3	-
SCAM 2005	19	7	-
SCAM 2006	22	9	2
SCAM 2007	23	7	1
SCAM 2008	29	14	-
SCAM 2009	20	10	-
SCAM 2010	21	15	1
SCAM 2011	21	9	1
SCAM 2012	22	12	4
SCAM 2013	24	12	-
SCAM 2014	35	16	1
SCAM 2015	30	18	?
Total	315	133	10

Tabela 4.2 Lista de ferramentas do SAMATE - NIST com código fonte não disponível

Ferramenta	Avaliacao
ABASH	código não disponível
ApexSec Security Console	código não disponível
Astrée	código não disponível
bugScout	código não disponível
C/C++test®	código não disponível
dotTEST™	código não disponível
Jtest®	código não disponível
HP Code Advisor (cadvise)	código não disponível
Checkmarx CxSAST	código não disponível
CodeCenter	código não disponível
CodePeer	código não disponível
CodeSecure	site offline
CodeSonar	código não disponível
Coverity SAVE™	código não disponível
Csur	código não disponível
DoubleCheck	código não disponível
Fluid	código não disponível
Goanna Studio and Goanna Central	código não disponível
HP QAInspect	código não disponível
Insight	código não disponível
ObjectCenter	código não disponível
Parfait	código não disponível
PLSQLScanner 2008	código não disponível
PHP-Sat	link para código offline
PolySpace	código não disponível
PREfix and PREfast	código não disponivel
QA-C, QA-C++, QA-J	código não disponível
Qualitychecker	código não disponível
Rational AppScan Source Edition	código não disponível
Resource Standard Metrics (RSM)	código não disponível
SCA	código não disponível
SPARK tool set	código não disponível
TBmisra®, TBsecure®	código não disponível
PVS-Studio	código não disponível
xg++	código não disponível

Tabela 4.3 Lista de ferramentas do SAMATE - NIST com código fonte disponível

Ferramenta	Avaliacao
BOON	código disponível
Clang Static Analyzer	código disponível
Closure Compiler	código disponível
Cppcheck	código disponível
CQual	código disponível
FindBugs	código disponível
FindSecurityBugs	código disponível
Flawfinder	código disponível
Jlint	código disponível
LAPSE	código disponível
Pixy	código disponível
PMD	código disponível
pylint	codigo disponivel
RATS (Rough Auditing Tool for Security)	código disponível
Smatch	código disponível
Splint	código disponível
UNO	código disponível
Yasca	código disponível
WAP	código disponível

Tabela 4.4 Lista com total de ferramentas a serem analisadas

Ferramenta	Linguagem	Fonte
BOON	ansic	industria
CQual	ansic	industria
RATS	ansic	industria
Smatch	ansic	industria
Splint	ansic	industria
UNO	ansic	industria
Clang Static Analyzer	cpp	industria
Cppcheck	cpp	industria
Jlint	cpp	industria
WAP	java	industria
Closure Compiler	java	industria
FindBugs	java	industria
FindSecurityBugs	java	industria
Pixy	java	industria
PMD	java	industria
Indus	java	academia
TACLE	java	academia
JastAdd	java	academia
WALA	java	academia
error-prone	java	academia
AccessAnalysis	java	academia
Bakar Alir	java/ada/python	academia
InputTracer	ansic	academia
srcML	cpp/cs (cs = C# ?)	academia
Source Meter	java	academia

CAPÍTULO 5

CONCLUSÃO

(à fazer)

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, L. T.; MIRANDA, J. M. de. Código limpo e seu mapeamento para métricas de código fonte. *Monografia de Graduação em Ciência da Computação*, p. 74, 2010. Disponível em: ⟨http://www.ime.usp.br/~cef/mac499-10/monografias/lucianna-joao/arquivos/monografia.pdf⟩.

BECK, K. *Implementation Patterns*. [S.l.]: Pearson Education, 2007.

BINKLEY, D. Source code analysis: A road map. In: IEEE COMPUTER SOCIETY. *2007 Future of Software Engineering*. [S.l.], 2007. p. 104–119.

CERT. *Secure Coding Tools*. 2016. [Online; acessado 23 Abril de 2016]. Disponível em: ⟨http://www.cert.org/secure-coding/tools/index.cfm⟩.

CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, IEEE, v. 20, n. 6, p. 476–493, 1994.

DARCY, D. P. et al. The structural complexity of software an experimental test. *Software Engineering, IEEE Transactions on*, IEEE, v. 31, n. 11, p. 982–995, 2005.

FEITELSON, D. G. From repeatability to reproducibility and corroboration. *ACM SIGOPS Operating Systems Review*, ACM, v. 49, n. 1, p. 3–11, 2015.

HARRISON, R.; COUNSELL, S. J.; NITHI, R. V. An evaluation of the mood set of object-oriented software metrics. *Software Engineering, IEEE Transactions on*, IEEE, v. 24, n. 6, p. 491–496, 1998.

HITZ, M.; MONTAZERI, B. *Measuring Coupling and Cohesion In Object-Oriented Systems*. [s.n.], 1995. Disponível em: ⟨http://www.isys.uni-klu.ac.at/PDF/1995-0043-MHBM.pdf⟩.

ISO, I. Iec25010: 2011 systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models. *International Organization for Standardization*, p. 34, 2011.

JúNIOR, M. R. P. Estudo de métricas de código fonte no sistema android e seus aplicativos. p. 82, 2015. Disponível em: ⟨https://fga.unb.br/tcc/software/tcc-2015.1-engenharia-de-software/marcos-ronaldo-pereira-junior/v3-tcc.pdf⟩.

KAZMAN, A. T. R. On the worthiness of software engineering research. 2016. Disponível em: ⟨http://shidler.hawaii.edu/sites/shidler.hawaii.edu/files/users/kazman/se_research_worthiness.pdf⟩.

KIRKOV, R.; AGRE, G. Source code analysis - an overview. *Cybernetics and Information Technologies*, v. 10, n. 2, p. 60–77, 2010.

KON, F. et al. Free and open source software development and research: Opportunities for software engineering. In: *SBES*. [s.n.], 2011. p. 82–91. Disponível em: ⟨http://dblp.org/db/conf/sbes/sbes2011.html\\#KonMLTCM11⟩.

KRISHNAMURTHI, S.; VITEK, J. The real software crisis: Repeatability as a core value. *Communications of the ACM*, ACM, v. 58, n. 3, p. 34–36, 2015.

MAFRA, S. N.; TRAVASSOS, G. H. Estudos primários e secundários apoiando a busca por evidência em engenharia de software. 2006.

MARTIN, R. C.; HAN, L. *Clean Code*. [S.l.]: Publishing House of Electronics Industry, 2012.

MCCABE, T. J. A complexity measure. *Software Engineering, IEEE Transactions on*, IEEE, n. 4, p. 308–320, 1976.

MCCONNELL, S. *Code Complete*. 2nd. ed. [S.l.]: Microsoft Press, 2004.

MEIRELLES, P. R. M. *Monitoramento de métricas de código-fonte em projetos de software livre*. Tese (Doutorado) — Universidade de São Paulo, São Paulo, Brazil, 2013.

NIST. *SAMATE - Source Code Security Analyzers*. 2016. [Online; acessado 20 Abril de 2016]. Disponível em: ⟨http://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html⟩.

PENG, R. D. Reproducible research in computational science. *Science (New York, Ny)*, NIH Public Access, v. 334, n. 6060, p. 1226, 2011.

PORTILLO-RODRíGUEZ, J. et al. Tools used in global software engineering: A systematic mapping review. p. 663–685, 2012. Disponível em: ⟨http://dblp.org/db/journals/infsof/infsof54.html\\#Portillo-RodriguezVPB12⟩.

PRLIć, A.; PROCTER, J. B. Ten simple rules for the open development of scientific software. *PLoS Computational Biology, vol. 8, issue 12, p. e1002802*, v. 8, p. 2802, dec 2012.

RIBEIRO, A. C. Análise estática de código-fonte com foco em segurança: Metodologia para avaliação de ferramentas. 2015.

ROSENBERG, L. H.; HYATT, L. E. Software quality metrics for object-oriented environments. *Crosstalk journal*, v. 10, n. 4, 1997.

SHARBLE, R. C.; COHEN, S. S. The object-oriented brewery: a comparison of two object-oriented development methods. *ACM SIGSOFT Software Engineering Notes*, ACM, v. 18, n. 2, p. 60–73, 1993.

SHIH, T. K. et al. Decomposition of inheritance hierarchy dags for object-oriented software metrics. In: *Engineering of Computer-Based Systems, 1997. Proceedings., International Conference and Workshop on*. [S.l.: s.n.], 1997. p. 238–245.

SOCIETY, I. C. *Guide to the Software Engineering Body of Knowledge*. Version 3.0. [S.l.], 2014.

SPIN. *Static Source Code Analysis Tools for C*. 2016. [Online; acessado 23 Abril de 2016]. Disponível em: ⟨http://www.spinroot.com/static⟩.

STOL, K.-J.; FITZGERALD, B. A holistic overview of software engineering research strategies. In: *3rd International Workshop on Conducting Empirical Studies in Industry*. [S.l.: s.n.], 2015. p. 8.

TERCEIRO, A. et al. Analizo: an extensible multi-language source code analysis and visualization toolkit. p. 6, 2010.

VITEK, J.; KALIBERA, T. Repeatability, reproducibility, and rigor in systems research. In: ACM. *Proceedings of the ninth ACM international conference on Embedded software*. [S.l.], 2011. p. 33–38.

VRIES, A. de. *Introducing the Reproducible R Toolkit and the checkpoint package*. 2014. Internet. [Online; Acessado em 24 de Maio de 2016]. Disponível em: ⟨http://blog.revolutionanalytics.com/2014/10/introducing-rrt.html⟩.

WHEELER, D. A. *Static analysis tools for security*. 2015. [Online; acessado 23 de Abril de 2016]. Disponível em: ⟨http://www.dwheeler.com/essays/static-analysis-tools.html⟩.

WIKIPEDIA. *List of tools for static code analysis*. 2016. [Online; acessado 23 Abril de 2016]. Disponível em: ⟨https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis⟩.