Universidade Federal da Bahia
Instituto de Matemática

Programa de Pós-graduação em Ciência da Computação

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

Joenio Marques da Costa
joenio@joenio.me

QUALIFICAÇÃO DE MESTRADO

Salvador
7 de maio de 2016

Universidade Federal da Bahia
Instituto de Matemática

Joenio Marques da Costa
joenio@joenio.me

CARACTERIZAÇÃO DA QUALIDADE INTERNA DE FERRRAMENTAS 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
7 de maio de 2016

AGRADECIMENTOS

DIGITE OS AGRADECIMENTOS AQUI

RESUMO

DIGITE O RESUMO AQUI

Palavras-chave: DIGITE AS PALAVRAS-CHAVE AQUI

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

(à fazer)

falar aqui de ferramentas de análise estática e o porque escolhi elas? artigo com um reumo geral de ferramentas de analise: Source Code Analysis: A Road Map.pdf

1.1 CONTRIBUIÇÕES ESPERADAS

(à fazer)

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 destes desafios, surge a Engenharia de Software, uma disciplina centrada no desenvolvimento de sistemas de software (WESSLéN, 2012) 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.

O crescimento no número de pesquisas e publicações em Engenharia de Software Experimental desperta a atenção para a necessidade de verificar a validade dos estudos empíricos realizados – um ponto central em qualquer pesquisa científica. A validade de um estudo empírico deve ser averiguada com o intuito de aumentar o nível de confiança em seus resultados, replicação costuma ser citado como um importante meio para atingir tal objetivo (ALMQVIST, 2006).

Um dos primeiros artigos discutindo replicação de experimentos em Engenharia de Software foi publicado por Basili et al. (MÄNTYLÄ; LASSENIUS; VANHANEN, 2010) e sugere replicação não apenas como uma escolha, mas como um possível "próximo passo"a ser tomado após o experimento original ser concluído. Apesar do conceito replicação de estudos empíricos em Engenharia de Software estar usualmente associado à experimentação, argumenta-se que ele deve ser estendido para incluir ao menos estudos de caso e surveys (BASILI; SELBY; HUTCHENS, 1986).

Em diversas linhas de pesquisa da Computação e, em especial, em Engenharia de Software, é bastante comum que novos sistemas de software sejam desenvolvidos, 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.

Softwares científicos são produtos de software e, em geral, precisam ser avaliados com uso de métodos científicos adequados, e, é de fundamental importância que estejam disponíveis e em funcionamento (KON et al., 2011). A disponibilidade destes softwares é peça fundamental para possibilitar a reprodutibilidade dos estudos relacionados, proporcionando assim, meios para validar tais pesquisas.

2.2 REPRODUTIBILIDADE

Reprodutibilidade (reproducibility) é a habilidade de replicar um experimento ou estudo em sua totalidade a fim de confirmar suas hipóteses, seja pelo autor ou por pesquisadores independentes. Este conceito é um ponto central do método científico e continua a receber bastante atenção ainda hoje, como pode ser verificado em estudos recentes.

Stodden (2009) preocupada com as barreiras legais para disponibilidade de artefatos de pesquisa propõe o framework "Reproducible Research Standard (RRS)", onde sugere formas de usar o licenciamento e as leis de copyright da melhor forma para manter disponíveis os produtos gerados durante pesquisas e assim viabilizar reprodutibilidade. Vitek e Kalibera (2011) em um estudo sobre reprodutibilidade e rigor científico destacam a importancia de se disponibilizar qualquer material suplementar gerado durante uma pesquisa de modo a possibilitar revisores verificarem e replicarem experimentos. Em 2012 um workshop intitulado "Reproducible Research: Tools and Strategies for Scientific Computing"(STODDEN, 2012) discutiu especificamente iniciativas e ferramentas voltadas a apoiar pesquisas reprodutíveis. Krishnamurthi e Vitek (2015) em um estudo sobre repetibilidade 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?". Stodden, Miguez e Seiler (2015) demonstram o projeto "ResearchCompendia.org", uma infraestrtura para reprodutibilidade e colaboração em ciência computacional. Além destes e tantos outros estudos em (HACKATHON, 2016) é possível acessar um guia sobre como desenvolver pesquisas científicas de forma que promovam a reprodutibilidade.

Apesar do termo reprodutibilidade ser relativamente concensual entre as várias áreas da ciência, existem alguns termos relacionados com uma certa diferença de significado, diante disto e preocupado em criar uma linguagem comum entre os pesquisadores Feitelson (2015) propõe as seguintes definições:

Repetição (repetition) Refazer exatamente o que outra pessoa fez usando os artefatos originais.

Replicação (replication) Replicar com precisão exatamente o que outra pessoa fez, recriando os artefatos.

Variação (variation) Repetir ou replicar exatamente o que aoutra pessoa fez, mas com alguma modificação controlada nos parâmetros.

Reprodução (reproduction) Recriar o espírito do que outra pessoa fez, usando seus próprios artefatos.

Corroboração (corroboration) Obter os mesmos resultados de outra pessoa, usando outros meios e procedimentos experimentais.

É conhecido que a ciência precisa de reprodutibilidade e corroboração para realmente fazer progressos mas a prática de forma abrangente ainda é um obstáculo, diante disso Peng (2011) sugere adotar soluções intermediárias, repetição, replicação, variação, e desta forma já teríamos uma grande melhoria sobre a situação atual onde muitos estudos em Engenharia de Software sofrem de dificuldades de repetição (KAZMAN, 2016).

Uma barreira comum para a reprodutibilidade, e consequentemente para repetição, replicação e variação é a indisponibilidade do código fonte. Toda pesquisa que possua qualquer processo computadorizado deve publicar seus códigos, eles precisam estar disponíveis, mesmo que os dados correspondentes não estejam, o código deve estar. Deacordo com o espectro (Figura 2.1) de reprodutibilidade a disponibilidade do código é o requisito mínimo.

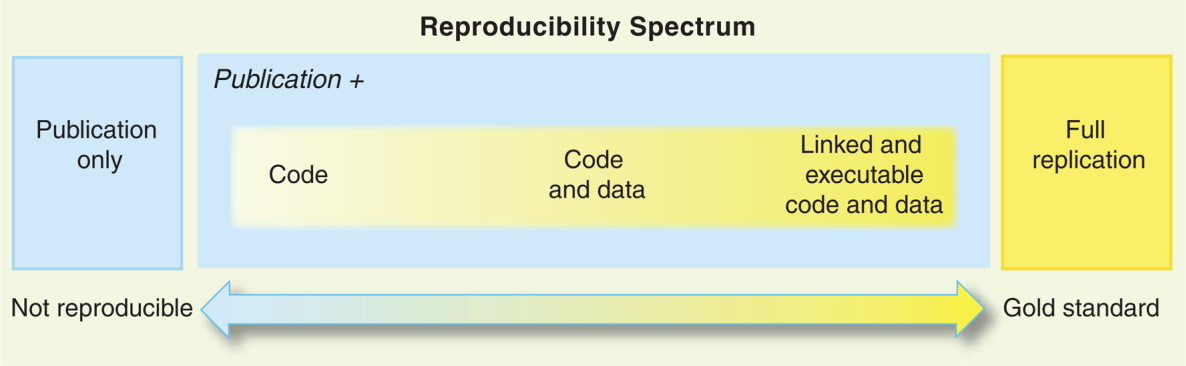

Figura 2.1 The spectrum of reproducibility(PENG, 2011)

Dentre as inúmeras iniciativas voltadas à reprodutibilidade de pesquisas científicas, destaco aqui o movimento chamado "Ciência Aberta"e a sua grande área "Open Reproducible Research"voltada a disseminar práticas que garantam a capacidade de replicação de estudos.

2.2.1 Ciência Aberta

Ciência Aberta é um movimento que tem por objetivo tornar a pesquisa científica, seus dados e sua disseminação acessíveis à todos os interessados, sejam amadores ou profissionais (WIKIPEDIA, 2015). Sua principal motivação está em possibilitar a reprodução dos resultados de pesquisas e em garantir transparência das metodologias utilizadas, isto aumenta o impacto social das pesquisas e gera economia de tempo e dinheiro para os pesquisadores e para as instituições (RIN/NESTA, 2010).

Este movimento é guiado por princípios básicos de transparência, acessibilidade e reusabilidade universais, disseminadas via ferramentas online, ele é dividido em quatro

grandes áreas: (1) Open Access, (2) Open Data, (3) Open Source e (4) Open Reproducible Research. Dentre elas destaca-se a Open Reproducible Research por preocupar-se com a reprodutibilidade dos resultados de pesquisas de forma independente (STODDEN, 2009) e aberta, no entanto, esta área tem recebido ainda pouca atenção da comunidade de pesquisa (PONTIKA et al., 2015) (GRAND et al., 2010b) apesar do aumento geral do interesse pelas práticas da Ciência Aberta (GRAND et al., 2010a).

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 – por exemplo, códigos fonte e dados – usualmente permanecem não publicados. Este é um problema sério já que um dos fundamentos da ciência é que novas descobertas sejam reproduzidas antes de serem consideradas parte da base de conhecimento (STODDEN, 2009).

Neste sentido, Prlić e Procter (2012) dão dicas para o desenvolvimento aberto de software científico e citam que disponibilizar o código criado durante pesquisas não apenas aumenta o impacto como também se torna essencial para outros reproduzirem os resultados encontrados. Eles citam ainda que manutenabilidade e disponibilidade do software após a publicação é o maior problema enfrentado pelos pesquisadores que desenvolvem tais softwares, e é aí que a participação no desenvolvimento aberto desde o início pode trazer maior benefício.

Dentro deste contexto, e considerando que pesquisas em engenharia de software produzem bastante softwares científicos, surge a preocupação de avaliar a qualidade de tais softwares a partir de métodos científicos adequados, especialmente em relação a sua manutenabilidade e disponibilidade.

2.3 QUALIDADE DE SOFTWARE

Qualidade de software é assunto presente em diversos estudos em engenharia de software, ela diz respeito à quão bem um software é projetado e o quanto este software está em conformidade com o projeto, embora existam inúmeras definições (KITCHENHAM; PFLEEGER, 1996) (MCCONNELL, 2004) (ISO, 2012) (WIKIBOOKS, 2013) (STAINES, 2015) pode-se afirmar 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 de um software.

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

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

Estas características, sejam internas ou externas, segundo a ISO/IEC 25010 (ISO, 2011) podem ser divididas em subcaracterísticas, usabilidade por exemplo é dividida

em: understandability, learnability, operability, attractiveness, compliance. Esta característica está relacionada a facilidade com que usuários aprendem e utilizam um sistema, ela passa por questões como facilidade de instalação, aprendizado, e uso. Wikibooks (2013) enumera algumas questões à respeito da usabilidade úteis para mensurar tal característica:

1. A interface de usuário é intuitica (auto-explicativa/auto-documentada)?
2. É fácil de executar uma operação simples?
3. É possível executar operações complexas?
4. O software dá mensagens de erro compreensíveis?
5. Os elementos se comportam como esperado?
6. O software é bem documentado?
7. A interface do usuário é responsiva ou muito lenta?

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 (correctness) e confiabilidade (reliability).

2.4 MÉTRICAS DE CÓDIGO-FONTE

Uma métrica, segundo definição da ISO/IEC 25010 (2001), é a composição de procedimentos para a definição de escalas e métodos para medidas (MEIRELLES, 2013).

Métricas são classificadas quanto aos critérios utilizados para determiná-las, quanto ao método de obtenção (MEIRELLES, 2013). Entre as classificações possíveis temos aquelas consideradas objetivas que tratam de características do código-fonte, muitos trabalhos tentam correlacionar métricas de software com qualidade de software. (Subramanyam e Krishnan, 2003) Briand et al. (2000) (Zuse, 1990).

Em suma, há uma variedade de métricas baseadas análise estática do código-fonte que permitem a avaliação de produtos de software (Gousios et al., 2007).

2.4.1 Análise de Código Fonte

O rápido crescimento no número de softwares nas últimas décadas leva a uma crescente demanda por mecanismos e ferramentas de apoio à compreensão de, desenvolvedores necessitam entender em profundade a implementação de um determinado software antes de realizar atividades de correção ou refatoração de forma eficiente (KIRKOV; AGRE, 2010).

Isto é evidenciado ao perceber que a complexidade dos softwares vem crescendo à cada dia (KIRKOV; AGRE, 2010), tornando assim, extremamente útil a existência de

ferramentas de análise automática de código-fonte, tais ferramentas auxiliam os desenvolvedores e engenheiros à compreender a implementação de um determinado software de forma profunda e abrangente.

Essas ferramentas geram modelos de alto nível representando entidades, relacionamentos, métricas, características ou outra informação qualquer extraída diretamente do código-fonte, e permitem que sejam geradas visualizações em diversos níveis de abstração representando o código-fonte.

Dentre as inúmeras sub-áreas da engenharia de software, este trabalho irá focar do domínio de ferramentas de análise de código-fonte, esta escolha tem por base o domínio do pesquisador nesta área e a escolha de um domínio é necessário para reduzir o escopo e viabilizar a revisão de estudos e ferramentas de um domínio específico, do contrário o trabalho seria muito extenso e a revisão sistemática seria inviável dentro do prazo de um trabalho de pesquisa num mestrado.

CAPÍTULO 3

METODOLOGIA

Neste capítulo será apresentada a metodologia utilizada no estudo como meio de validar as seguintes hipóteses:

H1: *Existem publicações sobre ferramentas de análise estática com disponibilidade de código-fonte*

H2: *Existem ferramentas de análise estática disponíveis livremente na indústria com disponibilidade de código-fonte*

H3: *Existem valores de referência para métricas de código-fonte para ferramentas de análise estática*

H4: *Ferramentas da indústria possuem melhores valores de métricas de código-fonte*

H5: *É possível relacionar a característica de qualidade externa usabilidade à valores de métricas de qualidade interna*

As seções à seguir descrevem as atividades de cada etapa da metodologia.

3.1 PLANEJAMENTO DO ESTUDO

3.1.1 Seleção das métricas

Meirelles (2013) realizou um estudo onde associou-se qualidade do produto de software à qualidade de código através de métricas de código-fonte como indicador para o sucesso de projetos de software livre, este estudo selecionou algumas métricas de código-fonte como base, a partir de critérios como existência de valores de referência para as métricas, além de X Y e B. Aqui deste trabalho utilizaremos como base às mesmas métricas:

3.1.1.0.1 Métricas de tamanho: LOC (Lines of Code), AMLOC (Average Method LOC), Total Number of Modules or Classes

3.1.1.0.2 Indicadores estruturais: NOA (Number of Attributes), NOM (Number of Methods), NPA (Number of Public Attributes), NPM (Number of Public Methods), ANPM (Average Number of Parameters per Method), DIT (Depth of Inheritance Tree), NOC (Number of Children), RFC (Response For a Class), ACCM (Average Cyclomatic Complexity per Method)

3.1.1.0.3 Métricas de acoplamento: ACC (Afferent Connections per Class), CBO (Coupling Between Objects), COF (Coupling Factor)

3.1.1.0.4 Métricas de coesão: LCOM (Lack of Cohesion in Methods), SC (Structural Complexity)

3.1.2 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 também 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 inicia-se a extração de seus atributos de qualidade interna.

3.1.3 Seleção da ferramenta de análise estática de código-fonte

Para realizar a caracterização das ferramentas através dos seus atributos de qualidade interna é necessário uma ferramenta capaz de analisar estaticamente o código-fonte destas ferramentas e extrair atributos relacionados à sua qualidade interna. Para isto utilizaremos o Analizo(KON, 2010). Falta Justificar! Quais vantagens? Referencias?

3.2 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.

3.2.1 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]

[1] http://ase-conferences.org

- 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.

3.2.2 Ferramentas da indústria

A seleção de ferramentas da indústria será feita a partir de uma busca manual em fontes encontradas na Internet sobre ferramentas de análise estática.

O projeto SAMATE[6] - *Software Assurance Metrics and Tool Evaluation* disponível em NIST (2016) mantém uma lista de ferramentas de análise estática mantida, mais sobre o projeto SAMATE 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, gratuitas mas não-livres 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, algums resultados preliminares podem ser encontrados nas Tabelas 4.3 e 4.2.

3.3 CARACTERIZAÇÃO DOS ARTIGOS

Caracterização dos papers analisados na revisão estruturada e caracterização teórica do ecosistema das ferramentas da academia.

3.4 CARACTERIZAÇÃO DAS FERRAMENTAS

Será realizada uma caracterização prática das ferramentas, tanto acadêmica quando da indústria, através da análise e extração de métricas de código-fonte das mesmas.

[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

3.5 EXEMPLO DE USO

Por fim, os valores de métricas de referência encontradas serão utilizadas como guia para refatorar a ferramenta Analizo.

Ler o TCC "Código Limpo e seu Mapeamento para Métricas de Código Fonte"onde foi feito trabalho similar.

(à fazer)

CAPÍTULO 4

CONCLUSÃO

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++ o Java que são suportadas pelo Analizo.

Ferramenta utilizada para isto foi a sloccount, uma ferramenta livre para contagem de linhas de código fonte, dá estatística de em qual linguagem é escrita em porcentagem. Abaixo destaco a linguagem de programação que tem maior porção em porcentagem.

Após analise ficamos com um total de 25 ferramentas, 15 da indústria e 10 da academia.

Segue na Tabela 4.4 a lista de todos os projetos e qual a fonte (industria ou academia):

4.2 CRONOGRAMA

(à fazer)

Tabela 4.1 Total de artigos analisados por edições do SCAM

Edição	Total de artigos	Artigos com ferramenta
SCAM 2001	23	-
SCAM 2002	18	-
SCAM 2003	21	-
SCAM 2004	17	-
SCAM 2005	19	-
SCAM 2006	22	2
SCAM 2007	23	1
SCAM 2008	29	-
SCAM 2009	20	-
SCAM 2010	21	1
SCAM 2011	21	1
SCAM 2012	22	4
SCAM 2013	24	-
SCAM 2014	35	1
SCAM 2015	?? (pendente)	?
Total	315	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

REFERÊNCIAS BIBLIOGRÁFICAS

ALMQVIST, J. P. F. Replication of controlled experiments in empirical software engineering - a survey. *Department of Computer Science, Faculty of Science, Lund University*, 2006.

BASILI, V. R.; SELBY, R. W.; HUTCHENS, D. H. Experimentation in software engineering. *Software Engineering, IEEE Transactions on*, v. 7, p. 733–743, 1986.

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

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

GRAND, A. et al. Muddying the waters or clearing the stream? open science as a communication medium. 2010.

GRAND, A. et al. On open science and public engagement with engineering. *European Association for Studies in Science and Technology*, p. 1–4, 2010.

HACKATHON rOpenSci. *Reproducibility Guide*. 2016. Internet. [Online; acessado em 06 de Maio 2016]. Disponível em: ⟨http://ropensci.github.io/reproducibility-guide/⟩.

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.

ISO, I. Iso/iec 25022:2012. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of quality in use*, 2012.

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.

KITCHENHAM, B.; PFLEEGER, S. L. Software quality: The elusive target. p. 12–21, 1996. Disponível em: ⟨http://dblp.org/db/journals/software/software13.html\#KitchenhamP96⟩.

KON, A. T. J. C. J. M. P. M. L. R. R. L. A. C. C. F. Analizo: an extensible multi-language source code analysis and visualization toolkit. p. 6, 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.

MÄNTYLÄ, M. V.; LASSENIUS, C.; VANHANEN, J. Rethinking replication in software engineering: Can we see the forest for the trees? In: *ICSE workshop RESER*. [S.l.: s.n.], 2010.

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.

PONTIKA, N. et al. Fostering open science to research using a taxonomy and an elearning portal. 2015.

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.

RIN/NESTA. *Open to All? Case studies of openness in research*. [S.l.], 2010. Disponível em: ⟨http://www.rin.ac.uk/our-work/data-management-and-curation/open-science-case-studies⟩.

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⟩.

STAINES, A. S. Some aspects of software quality assurance for small projects. *International Journal of Scientific and Engineering Research*, v. 6, n. 5, p. 441–445, 2015.

STODDEN, V. Enabling reproducible research: Open licensing for scientific innovation. *International Journal of Communications Law and Policy, Forthcoming*, v. 13, p. 1–25, 2009.

STODDEN, V. Reproducible research: Tools and strategies for scientific computing. *Computing in Science & Engineering*, IEEE, n. 4, p. 11–12, 2012.

STODDEN, V.; MIGUEZ, S.; SEILER, J. Researchcompendia.org: Cyberinfrastructure for reproducibility and collaboration in computational science. *Computing in Science & Engineering*, AIP Publishing, v. 17, n. 1, p. 12–19, 2015.

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.

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.

WESSLéN, C. W. P. R. M. H. M. C. O. B. R. A. *Experimentation in Software Engineering*. [S.l.]: Springer Science & Business Media, 2012.

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⟩.

WIKIBOOKS. *Introduction to Software Engineering*. [S.l.]: Wikibooks, 2013.

WIKIPEDIA. *Open Science*. 2015. Open Science. [Online; acessado 13 Outubro de 2015]. Disponível em: ⟨http://en.wikipedia.org/wiki/Open_science⟩.

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⟩.