Universidade Federal da Bahia
Instituto de Matemática

Programa de Pós-graduação em Ciência da Computação

CARACTERIZAÇÃO DA COMPLEXIDADE ESTRUTURAL EM FERRAMENTAS DE ANÁLISE ESTÁTICA DE CÓDIGO-FONTE

Joenio Marques da Costa
joenio@joenio.me

QUALIFICAÇÃO DE MESTRADO

Salvador
08 de Julho de 2016

Universidade Federal da Bahia
Instituto de Matemática

Joenio Marques da Costa
joenio@joenio.me

CARACTERIZAÇÃO DA COMPLEXIDADE ESTRUTURAL EM 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
08 de Julho de 2016

AGRADECIMENTOS

(pendente)

RESUMO

(pendente)

Palavras-chave:
(pendente)

ABSTRACT

(pendente)

Keywords:
(pendente)

SUMÁRIO

Capítulo 1—Introdução 1

1.1 Objetivos 1
1.2 Contribuições esperadas 1

Capítulo 2—Análise estática de código-fonte 3

2.1 Anatomia da análise de código-fonte 4
2.1.1 Extração de dados 4
2.1.2 Representação interna 5
2.1.3 Análise da representação interna 5

Capítulo 3—Métricas de código-fonte 7

3.1 Complexidade estrutural 7

Capítulo 4—Metodologia 9

4.1 Trabalhos relacionados 9
4.2 Hipóteses 9
4.3 Planejamento do estudo 9
4.3.1 Seleção de métricas 9
4.3.2 Seleção de ferramentas de análise estática 10
4.3.3 Revisão estruturada 11
4.4 Coleta de dados 11
4.4.1 Ferramentas da academia 11
4.4.2 Ferramentas da indústria 11
4.5 Análise de dados 12
4.5.1 Caracterização dos artigos 12
4.5.2 Caracterização das ferramentas 13
4.5.3 Distribuição dos valores das métricas 13
4.5.4 Cálculo de distância e modelo de aproximação 13

Capítulo 5—Caracterização dos artigos 15

Capítulo 6—Caracterização das ferramentas 17

6.1 Resultados 17

Capítulo 7—Evolução de uma ferramenta da análise estática 19

Capítulo 8—Conclusão 21

8.1 Limitações do trabalho 21
8.2 Trabalhos futuros 21

LISTA DE FIGURAS

2.1 CodeSonar: Static Analysis Representation 4

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

(pendente)

1.1 OBJETIVOS

O objetivo principal deste trabalho é compreender as ferramentas de software para análise estática de código-fonte, do ponto de vista de sua manutenabilidade, a partir da análise de sua complexidade estrutural, discutindo quais características arquiteturais explicam seus atributos de qualidade interna.

Objetivos específicos:

- Realizar uma revisão da literatura para caraterizar a arquitetura das ferramentas de análise estática de código-fonte.
- Realizar uma revisão estruturada, com base em artigos publicados em conferências relacionadas, para selecionar e obter código-fonte de ferramentas de análise estática (explicadas na Seção 4.3.3), para coletar suas métricas de código-fonte.
- Selecionar e obter o código-fonte de ferramentas de análise estática desenvolvidas pelo indústria (explicadas na Seção 4.4.2), para coletar suas métricas de código-fonte.
- Propor intervalos de referência para a observação parametrizada da qualidade interna das ferramentas de análise estática, a partir de suas métricas de código-fonte.
- Cálculo da distância bayesiana entre valores de referência e os valores das ferramentas estudadas
- Evoluir uma ferramenta de análise de código-fonte para coleta de métricas de código-fonte (detalhado no Capítulo 7).
- Um conjunto de estratégias para a manutenção e evolução de ferramentas de análise estática de código-fonte.

1.2 CONTRIBUIÇÕES ESPERADAS

(pendente)

CAPÍTULO 2

ANÁLISE ESTÁTICA DE CÓDIGO-FONTE

Análise estática de código-fonte é um ramo da engenharia de software voltado para obter informações acerca de um programa a partir do seu código-fonte sem necessidade de execução, e sem requerer qualquer artefato além do próprio código. É uma atividade meio voltada a atingir diversos fins em uma variedade de tarefas comuns da engenharia de software, muitas dessas tarefas são substancialmente úteis em atividades de manutenção.

Segue uma lista extensa de tarefas que podem usualmente ser suportadas por análise estática segundo Binkley (2007):

- architecture recovery
- assertion discovery
- automotive software engineering
- clone detection
- comprehension
- debugging
- empirical software engineering research
- fault location
- middleware
- model checking in formal analysis
- model-driven development
- optimization techniques in software engineering
- performance analysis
- program evolution
- quality assessment
- reverse engineering
- safety critical
- software maintenance
- software reliability engineering
- software versioning
- specification semantics
- symbolic execution
- testing
- tools and environments
- validation (conformity checking)
- verification, sound formal
- verification, unsound syntactic
- visualizations of analysis results
- web application development

Seja em qual atividade a análise estática esteja sendo aplicada ela tem uma importância significativa pois o código-fonte de um programa é considerado como "a verdade" sobre o que ele faz ou significa. Apenas através do código-fonte podemos responder com precisão a pergunta: "O que é que este programa significa?".

2.1 ANATOMIA DA ANÁLISE DE CÓDIGO-FONTE

Cruz, Henriques e Pinto (2009), Binkley (2007) realizam ambos um estudo onde definem a estrutura comum da análise de código e afirmam que esta estrutura é organizada em três componentes: a) extração de dados, b) representação interna, e c) análise da representação interna. Um diagrama exemplificando esta estrutura pode ser visto na Imagem 2.1.

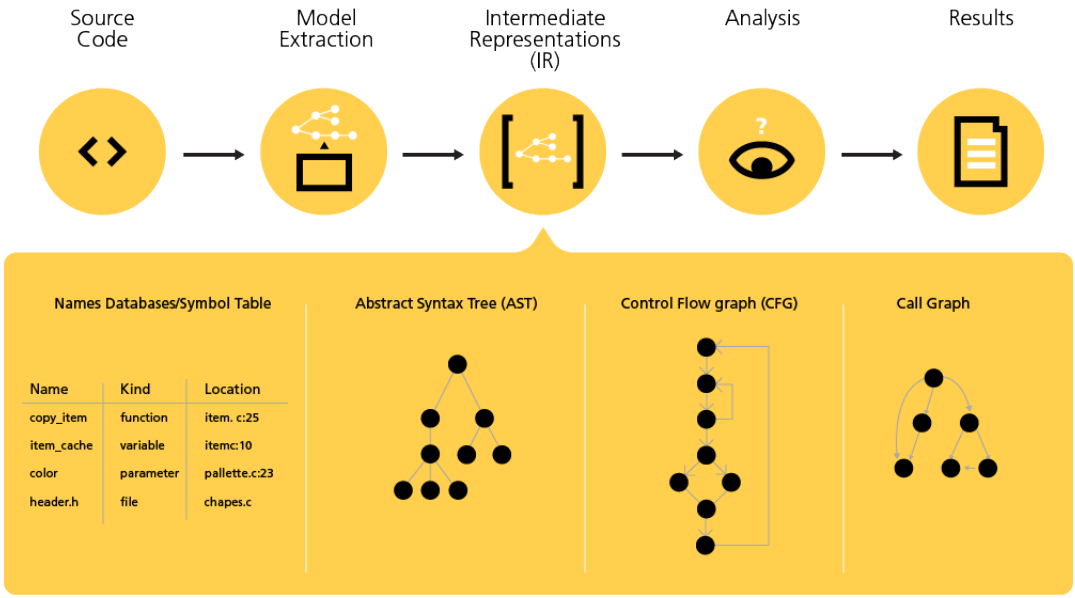

Figura 2.1: CodeSonar: Static Analysis Representation

As seções à seguir detalham cada componente da análise de código-fonte.

2.1.1 Extração de dados

O processo de recuperar dados para futuro processamento ou armazenamento é chamado de extração de dados, exportar estes dados em uma representação intermadiária é uma estratégia comum para facilitar análise e transformação de dados e possivelmente adição de metadados.

O primeiro componente da análise de código-fonte é a extração de dados, responsável por processar o código-fonte e transformar em uma ou mais representações internas, em essência este componente converte a sintaxe de um programa em uma outra sintaxe abstrata e mais adequada para análise posterior.

Este componente é usualmente chamado de analisador sintático (ou *parser*) e a tarefa realizada por ele é considerada um mal necessário da análise de código-fonte. Apesar de teoricamente não ser uma tarefa difícil, a complexidade das linguagens de programação modernas, podem restringir significativamente a análise de código-fonte.

2.1.2 Representação interna

Após extração de dados do código-fonte, é necessário representar isto em um forma mais abstrata, esta é a responsabilidade do segundo componente da análise de código-fonte: armazenar os dados coletados usando uma representação interna em um formato mais adequada para análise automática, o principal papel deste componente é abstrair aspectos particulares do programa.

Muitas das representações internas existentes tem seu surgimento na área de compiladores e algumas delas são produzidas diretamente pelo *parser* enquanto outras requerem uma análise específica. O formato mais comum é baseado em grafos, dentre os quais o mais amplamente utilizado é Control Flow Graph (CFG), ou Call Graph.

Além deste existe atualmente um gama de formatos diferentes:

- Identifiers Table
- Abstract Syntax Tree (AST)
- Decorated Abstract Syntax Tree (AST)
- Control Flow Graph (CFG)
- Value Dependence Graph (VDG)
- Call Graph
- Module Dependence Graph (MDG)
- Trace Flow Graph (TFG)
- Static Single Assignment (SSA)

Estas representações costumam ser capaz de serem utilizadas seja com a abordagem de análise estática e dinâmica, e suas variações passando por grafos eoutros formatos são usados de acordo com o tipo de análise e seu propósito. Numa aplicação real é comum combinar diferentes tipos de grafos ou AST com Identifier Tables no sentido de enriquecer e estruturas a informação extraída.

2.1.3 Análise da representação interna

Após organizar os dados extraídos em uma representação intermediária ou transformar isto em informação, este terceiro componente da análise de código é responsável por realizar inferências de conhecimento, este processo requer que as informações armazenadas estejam interconectadas e também interrelacionadas com conhecimento anterior. Esta análise pode gerar conhecimento quantitativo ou qualitativo, como por exemplo métricas de software ou mineração de dados, respectivamente. Importante lembrar que técnicas de visualização são cruciais para a efetividade deste processo.

Este componente realizar a análise pripriamente dita e pode ser classificada em seis dimensões básicas: estática versus dinâmica, sound versus unsound, segura vesus insegura, sensível ao fluxo versus insensível ao fluxo, sensível ao contexto versus insensível ao contexto, e, em relação à sua complexidade.

Estas características da análise vão afetar o resultado gerado quanto à confiabiliade, completude, garantia de precisão, eficiência, performance, complexidade computacional, precisão da informaçao, dentre outras caractarísticas do resultado da análise de código.

CAPÍTULO 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.

3.1 COMPLEXIDADE ESTRUTURAL

(pendente) “apresentar e definir SC pois faremos um gancho com trabalho de terceiro onde caracterizou projetos utilizando apenas complexidade estrutural.”

CAPÍTULO 4

METODOLOGIA

Visando então caracterizar ferramentas de análise estática de código-fonte será feito um levantamento de artigos com publicação de ferramentas deste domínio. Além das ferramentas encontradas a partir das publicações 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 ferramentas com disponibilidade de código-fonte a partir das publicações.

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.

4.1 TRABALHOS RELACIONADOS

(pendente)

4.2 HIPÓTESES

Para guiar os estudos, conforme os objetivos acima, definimos as seguintes hipóteses:

H1: *É possível calcular valores de referência de métricas de código-fonte para ferramentas de análise estática a partir de um conjunto de softwares da academia e da indústria*

H2: *Ferramentas de análise estática tendem a ter uma maior complexidade estrutural do que ferramentas de outros domínios de aplicação*

H3: *Dentre as ferramentas de análise estática de código-fonte, aquelas desenvolvidas na indústria apresentam uma menor complexidade estrutural*

(pendente) "explicar as hipóteses aqui."

4.3 PLANEJAMENTO DO ESTUDO

4.3.1 Seleção de 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.

(pendente) "retomar complexidade estrutural e linkar com trabalho de terceiro, iremos olhar com mais detalhes algumas ferramentas e para este subconjunto vamos analisar a organização arquitetural dos módulos, isto será base para o guia de refatoração que faremos."

4.3.2 Seleção 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 da base mantida pelo projeto SAMATE... (pendente) "falar e explicar NIST e seu projeto SAMATE e o porque escolhemos esta fonte."

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.

4.3.3 Revisão estruturada

(pendente)

4.4 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.

4.4.1 Ferramentas da academia

A seleção de ferramentas será relizada através de uma revisão estruturada dos artigos selecionados a partir da conferência SCAM - Source Code Analysis and Manipulation Working Conference[1] e mais uma dentre as seguintes conferências:

- ASE - Automated Software Engineering[2]
- CSMR[3] - Conference on Software Maintenance and Reengineering[4]
- ICSME - International Conference on Software Maintenance and Evolution[5]

4.4.2 Ferramentas da indústria

A seleção de ferramentas da indústria será feita de forma não estruturada a partir de uma busca livre e manual encontradas no site do 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).

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

Esta fonte será pesquisada manualmente em busca de ferramentas de análise estática que tenham sido desenvolvidas no contexto da indústria, alguns resultados preliminares podem ser encontrados nas Tabelas...

4.5 ANÁLISE DE DADOS

(pendente)

4.5.1 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 ferramenta , 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 serão 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.

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

4.5.2 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.

(pendente)

4.5.3 Distribuição dos valores das métricas

(pendente)

4.5.4 Cálculo de distância e modelo de aproximação

(pendente)

CAPÍTULO 5

CARACTERIZAÇÃO DOS ARTIGOS

(pendente) “documentar aqui os artigos incluídos na revisão estruturada, identificar aqueles que publicam ferramenta, dentre os que publicam quais tinham o software de fato disponível, as referencias indicadas no artigo para obtenção do software estavam corretas? foi necessário contactar o autor?”

CAPÍTULO 6

CARACTERIZAÇÃO DAS FERRAMENTAS

(pendente) “métricas das ferramentas, cálculo dos percentis, score de aproximação, valores de referências, discussão sobre cada métrica, discussão sobre as ferramentas com melhor qualidade, dentre as selecionadas como melhores discutir detalhes da sua arquitetura.”

6.1 RESULTADOS

(pendente)

CAPÍTULO 7

EVOLUÇÃO DE UMA FERRAMENTA DA ANÁLISE ESTÁTICA

(pendente) “a caracterização das ferramentas dará indícios para documentar sugestões de refatoração para ferramentas de análise estática, estas sugestões serão aplicadas na refatoração do Analizo.”

CAPÍTULO 8

CONCLUSÃO

(pentende) “discutir as contribuições dando resposta ao que foi colocado na introdução.”

8.1 LIMITAÇÕES DO TRABALHO

(pendente)

8.2 TRABALHOS FUTUROS

(pendente)

REFERÊNCIAS BIBLIOGRÁFICAS

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

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.

CRUZ, D. d.; HENRIQUES, P. R.; PINTO, J. S. Code analysis: Past and present. 2009.

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.

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⟩.

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

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

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⟩.

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.

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