EUDES DE CASTRO LIMA

UMA ANÁLISE DOS VALORES DE REFERÊNCIA DE ALGUMAS MEDIDAS DE *SOFTWARE*

LAVRAS - MG
2014

EUDES DE CASTRO LIMA

UMA ANÁLISE DOS VALORES DE REFERÊNCIA DE ALGUMAS MEDIDAS DE SOFTWARE

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Banco de Dados e Engenharia de *Software*, para a obtenção do título de Mestre.

Orientador
Dr. Antônio Maria Pereira de Resende
(DCC/UFLA)

LAVRAS – MG
2014

Ficha Catalográfica Elaborada pela Coordenadoria de Produtos e Serviços da Biblioteca Universitária da UFLA

Lima, Eudes de Castro.
Uma análise dos valores de referência de algumas medidas de software / Eudes de Castro Lima. – Lavras : UFLA, 2014.
181 p. : il.

Dissertação (mestrado) – Universidade Federal de Lavras, 2014.
Orientador: Antônio Maria Pereira de Resende.
Bibliografia.

1. Software. 2. Software - Qualidade. 3. Análise comparativa. I. Universidade Federal de Lavras. II. Título.

CDD – 005.14

EUDES DE CASTRO LIMA

UMA ANÁLISE DOS VALORES DE REFERÊNCIA DE ALGUMAS MEDIDAS DE SOFTWARE

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Banco de Dados e Engenharia de *Software*, para a obtenção do título de Mestre.

APROVADA em 11 de Junho de 2014.

Dr. André Pimenta Freire	DCC/UFLA
Dr. Heitor Augustus Xavier Costa	DCC/UFLA
Dra. Kécia Aline Marques Ferreira	DECOM/CEFET

Dr. Antônio Maria Pereira de Resende
(Orientador - DCC/UFLA)

LAVRAS – MG
2014

À minha mãe Creusa,
meu pai Érico (In memorian),
meus irmãos Elvis e Érico Júnior.

DEDICO

AGRADECIMENTOS

Agradeço à Universidade Federal de Lavras (UFLA), por mais uma vez, abrir as portas para que pudesse realizar este sonho.

Agradeço a CAPES, pelo incentivo e suporte financeiro, que foi de fundamental importância, para o desenvolvimento deste trabalho.

Ao professor Dr. Antônio Maria Pereira de Resende, professor, orientador e amigo, pelas orientações, pelas palavras de incentivo, pelos ensinamentos, por acreditar e, principalmente, por não me deixar desistir.

Aos professores do Departamento de Ciência da Computação (DCC), em especial, aos professores Dr. Wilian Soares Lacerda, Dr. Heitor Augustus Xavier Costa e Dr. André Pimenta Freire, pela paciência, sugestões e ensinamentos transmitidos.

Agradeço aos colegas do Grupo de Pesquisa em Engenharia de *Software* (PqES), Cibely Cobo, Ana Paula Martins, Fernando Tomazelli, Danilo Batista, Bruno Rosa, José Henrique. A ajuda de vocês foi essencial para este trabalho.

À minha mãe Creusa e irmãos Érico e Elvis, por todo suporte que sempre me propiciaram, por serem os alicerces e pilares, por nunca duvidarem e estarem sempre presente, nos momentos bons ou ruins. A meus primos Diogo e Danilo pela convivência e pela paciência. E, também, não menos importante, minha avó e avô (*in memorian*), tios, cunhadas Aramália e Jussara e meu querido afilhado, Igor. Muito Obrigado.

Aos amigos e colegas do mestrado Karen Bezerra, Rodrigo Amador, Christiane Faleiro, Claudiane Oliveira, Ramon Abílio, Frederico Santos, Vladmir Piccolo, pelo apoio, incentivo, companheirismo, pelas risadas, por tudo.

À Tatiana Cantelle, por me ajudar a recuperar a motivação em um momento delicado, pelas palavras, pela paciência, preocupação, carinho e pelos momentos juntos.

Ninguém vence sozinho...
Obrigado a todos!

“É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória, nem derrota.”

Theodore Roosevelt

RESUMO

As medições de software exercem um papel fundamental nas organizações, propiciando medidas para características de software. É por meio dessas medidas que um engenheiro de software avalia o estado atual do software e diagnostica se o projeto encontra-se dentro do esperado ou há um desvio em sua condução. No entanto, medidas poderiam ser mais utilizadas, se houvesse valores de referência confiáveis associados a elas. Alguns autores vêm propondo valores de referência, os quais não têm sido amplamente aceitos, pois utilizam técnicas pouco confiáveis, ou não foram validados adequadamente, ou se restringem a um contexto, como: domínio de aplicação, região, ou grupo de empresas estudadas, dentre outros. A fim de reduzir esta deficiência que restringe o uso de medidas, este trabalho foi realizado com o objetivo de realizar uma análise comparativa entre os valores de referência das principais medidas de software apresentadas na literatura científica e os valores praticados atualmente por desenvolvedores de software. Os valores de referência da literatura foram obtidos, por meio da leitura de 19 artigos científicos identificados por uma Revisão Sistemática da Literatura. Os valores praticados no mercado foram obtidos por uma coleta de dados e análise estatística de 10 medidas, em 107 sistemas de software, sendo cada software amostrado em 3 versões distintas, totalizando 321 sistemas de software analisados. Ao término das análises, uma análise comparativa foi realizada para cada medida selecionada.

Palavras Chave: Revisão Sistemática da Literatura; Valores de referência; Medidas de *Software;* Análise Comparativa; Qualidade de *Software*.

ABSTRACT

Software measurements exercise a fundamental role in the organizations, providing measures for software characteristics. It is by means of these measures that a software engineer evaluates the current state of the software and diagnoses if the project is inside the expected of if there is a deviation in its conduction. However, measures might be more used if trustworthy reference values were presented in association. A few authors have been proposing reference values, which have not been widely accepted, since they use little trustworthy techniques, or were not adequately validated, or they are restricted to a context, such as: application domain, region, or group of enterprises studied, among others. In order to reduce this deficiency, which restricts the use of measures, this work was conducted with the objective of performing a comparative analysis between the reference values of the main software measures presented in scientific literature and the values currently practiced by software developers. The reference values from literature were obtained, by means of reading 19 scientific articles identified by a Systematic Literature Review. The values practiced in the market were obtained by data collection and statistical analysis of 10 measures, in 107 software systems, each software being sampled in 3 distinct versions, totalizing 321 software systems analyzed. At the end of the analyses, a comparative analysis was performed for each selected measure.

Key Words: Systematic Review of Literature; Reference values; Measures of *Software;* Comparative Analysis; *Software* Quality.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

AHF – *Attribute Hiding Factor*

AIF – *Attribute Inheritance Factor*

BCa – *Biased Corrected Accelerated*

CBO – *Coupling Between Object Classes*

CC – *Cyclomatic Complexity*

CK – *Chidamber e Kemerer*

CLF – *Clustering Factor*

COF – *Coupling Factor*

DIT - *Depth Inheritance Tree*

FP - *Function Points*

LCOM – *Lack of Cohesion in Methods*

LK – *Lorenz e Kidd*

LOC – *Lines of Code*

MHF – *Method Hiding Factor*

MIF – *Method Inheritance Factor*

MOOD – *Metrics for Object Oriented Design*

nCl – *Number of Class*

NOC – *Number of Children*

NOM – *Number of Methods*

OO – *Object Oriented*

PF – *Pholimorphism Factor*

QMOOD – *Quality Model for Object Oriented Design*

RF – *Reuse Factor*

RFC – *Response for Class*

ROC – *Receiver Operating Characteristc*

RSL – Revisão Sistemática da Literatura

UCP – *Use Case Point*

WMC - *Weight Methods per Class*

SUMÁRIO

1 INTRODUÇÃO

As medições de *software* exercem papel importante nas organizações, pois elas propiciam medidas para características de *software*, como: manutenibilidade, portabilidade, inteligibilidade, corretitude, dentre outros. Essas medidas possibilitam aos engenheiros, arquitetos e gerentes de projeto de *software* avaliar o estado atual do *software*, a fim de diagnosticar se o projeto, o produto ou o processo, encontram-se dentro do esperado ou há um desvio em sua condução.

Ao longo dos anos, uma grande variedade de medidas de *software* (ABREU; CARAPUÇA, 1994; ALBRECHT, 1979; BIEMAN; KANG, 1995; CHIDAMBER; KEMERER, 1994; DALLAL; BRIAND, 2012; GUI; SCOTT, 2008; HALSTEAD, 1977; HENRY; KAFURA, 1981; LORENZ; KIDD, 1994; MCCABE, 1976; WASHIZAKI; YAMAMOTO; FUKAZAWA, 2003) e ferramentas automatizadas de medição (CCCC..., 2012; MCCABE IQ, 2012; TERCEIRO et al., 2010) tem sido propostas. No entanto, apesar da importância das medidas de *software*, elas não têm sido amplamente aplicadas na indústria (FENTON; NEIL, 1999; LANZA; MARINESCU, 2006). Acredita-se que uma das razões seja a falta de valores de referência para a maioria das medidas (TEMPERO, 2008).

Valores de referência para medidas de *software* são de suma importância, pois fornecem um conjunto de valores utilizados para interpretar os resultados de uma medição. É por meio das comparações de medidas, com seus respectivos valores de referência, que engenheiros, arquitetos e gerentes de projeto de *software* verificam se o projeto está acima de um padrão desejado ou, se analisando no tempo, está melhorando, piorando ou estável.

Em alguns casos, os valores de referência são conhecidos, mas não amplamente aceitos. Isso gera incerteza que, de acordo com Lanza e Marinescu (2006), inibe a popularização das medidas de *software*.

Nas últimas décadas, autores como Alves, Ypma e Visser (2010), Coppick e Cheatham (1992), Ferreira et al. (2011), Rosenberg, Stapko e Gallo (1999) e Shatnawi et al. (2010) vêm propondo técnicas e valores de referência para algumas medidas de *software*. Há artigos como o de Coppick e Cheatham (1992), French (1999) e Rosenberg, Stapko e Gallo (1999), que apresentam valores de referência baseados em "experiência própria" (conhecimento tácito) sem nenhuma análise estatística ou técnica que suporte a afirmação. No entanto, uma vez obtidos, em um contexto específico ou sem técnica adequada, os valores de referência publicados tendem a não ser generalizáveis para além do contexto de sua obtenção.

A medição de *software* está diretamente ligada à garantia de qualidade. Por meio da medição, o produto de *software* pode ser monitorado, proporcionando avaliação constante e possibilitando ajustes com base nos desvios detectados. Todavia, esses ajustes devem ser realizados com base nos valores de referência.

Há uma situação similar na medicina. Quando um exame de sangue é feito, os valores obtidos no exame são comparados com seus respectivos valores de referência. Com base na comparação realizada, o médico faz o diagnóstico e define o tipo e a dose do remédio a ser tomado, caso haja anormalidade nos resultados.

Na medicina, há valores de referência para grande parte dos exames, permitindo o diagnóstico de pacientes. Porém, na Engenharia de *Software*, ainda há longa jornada para obtenção desses valores. De acordo com Ferreira et al. (2009), a obtenção de valores de referência de medidas de *software* é uma

questão em aberto e sua solução constitui uma contribuição importante para viabilizar o uso efetivo de medidas no desenvolvimento de *software*.

Além disso, foi constatado que a comunidade científica e tecnológica não dispõe de uma análise atualizada sobre as medidas de *software* e a confiabilidade de seus valores de referência, o que dificulta, ainda mais, a aplicação das medidas.

Considerando a necessidade da comunidade científica e tecnológica, os objetivos deste trabalho foram definidos.

1.1 Objetivo Geral

Realizar uma análise comparativa entre os valores de referência das principais medidas de *software*, apresentadas na literatura científica e os valores praticados atualmente pelos desenvolvedores de *software*.

1.2 Objetivos Específicos

Para a consecução do objetivo proposto, as seguintes ações foram realizadas:

- realizar uma Revisão Sistemática da Literatura (RSL) e identificar medidas de *software* que possuem valores de referência associados a elas;
- analisar os valores de referência identificados na literatura, a fim de verificar sua confiabilidade e sugerir novos valores com base nos valores existentes;

- fazer a coleta de medidas em sistemas de livres, disponíveis na internet e em bases de dados;
- realizar análise estatística sobre os dados coletados;
- realizar a análise dos valores praticados atualmente pelos desenvolvedores de *software*;
- comparar os dados identificados na RSL e os valores praticados no mercado.

1.3 Estrutura do Trabalho

O presente trabalho encontra-se estruturado em 7 capítulos, sendo o primeiro uma breve introdução, que contém uma contextualização, definição do problema em estudo, o objetivo geral e os objetivos específicos.

No Capítulo 2, apresenta-se o Referencial Teórico em que são fundamentados os principais conceitos utilizados neste trabalho.

No Capítulo 3, apresenta-se a metodologia onde são descritas as atividades realizadas.

No Capitulo 4, apresenta-se a aplicação da Revisão Sistemática da Literatura.

No Capítulo 5, apresentam-se a coleta e a análise estatística dos dados.

No Capítulo 6, são apresentados os resultados de uma análise comparativa entre as medidas identificadas com a aplicação da Revisão Sistemática da Literatura e as práticas de mercado de *software* livre.

Por fim, no Capítulo 7, são apresentadas as principais conclusões obtidas neste trabalho, as contribuições, limitações e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste Capítulo, são descritos os conceitos básicos necessários para o entendimento deste trabalho.

2.1 Qualidade de *Software*

Atingir a alta qualidade tornou-se meta de diversas organizações, pois conforme Sommerville (2011), desenvolver e entregar produtos com baixa qualidade não é mais aceitável nos dias atuais.

De acordo com Kan (2002), definir qualidade de *software* não é simples, em razão da subjetividade do termo “qualidade” e por causa da intangibilidade do *software*. Para Sommerville (2011), o conceito de qualidade do *software* é complexo, pois não é diretamente comparável com a qualidade na manufatura.

De acordo com Pressman (2011, p. 360), a qualidade de *software* pode ser definida como “uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam”.

Para tratar as necessidades de qualidade do produto de *software*, existem normas internacionais que identificam aspectos relevantes sobre a mensuração dessa qualidade. Como exemplo, a norma ISO/IEC 25000 (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION - ISO, 2014).

A norma ISO/IEC 25000 (ISO, 2014), também, conhecida como SQuaRE - (*Quality Requirements and Evaluation*), é o resultado da revisão da ISO/IEC 9126 (ISO, 1991) e da ISO/IEC 14598 (ISO, 1999). Algumas alterações presentes na norma ISO/IEC 25000 (ISO, 2014) são:

- modificações nos modelos de qualidade: o modelo de qualidade para Qualidade Externa e Interna, presente na ISO/IEC 9126 (ISO,

1991), foi harmonicamente combinado em um único modelo denominado Qualidade de Produto;

- alterações em algumas características e subcaracterística: a ISO/IEC 25000 (ISO, 2014) define um modelo de qualidade de produto considerando 8 características: adequação funcional, eficiência de desempenho, compatibilidade, confiabilidade, segurança, manutenibilidade, portabilidade e usabilidade;
- substituição de alguns termos: por exemplo: o termo “métrica” pelo termo “medição”.

2.2 Medidas de *Software*

De acordo com Pressman (2011), medir é fundamental em qualquer disciplina de Engenharia, e a Engenharia de *Software* não é exceção. Medidas são essenciais para quantificar e entender as características de um *software*.

Existem vários motivos para medir um *software*, como: indicar a qualidade do produto, avaliar a produtividade e formar um histórico de estimativas. Além disso, as medidas exercem papel fundamental na fase de planejamento, pois por meio delas, é possível entender, avaliar, controlar e monitorar produtos, processos ou projetos. Conforme Fenton e Neil (1999), as medidas fornecem informações que auxiliam, principalmente, na tomada de decisão.

De acordo com Fenton e Pfleeger (1997), a medição é o processo de atribuir números a atributos de entidades no mundo real de tal forma a descrevê-los de acordo com regras claramente definidas. A maneira de mensurar o *software* é utilizando medidas de *software.*

Uma métrica, conforme IEEE Std 610.12 (INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS - IEEE, 1990, p. 47), é

“uma medida quantitativa do grau em que um sistema, componente ou processo possui um determinado atributo”. Segundo a definição da norma ISO/IEC 9126 (ISO, 1991, p. 2), uma métrica é “um método e escalas de medições definidos”. Ferreira (2013) define métrica como sendo um conjunto de regras que presidem a avaliação de um atributo de algo relacionado ao *software*. Para Sommerville (2011), métrica de *software* consiste em qualquer tipo de medição que se refere a um sistema de *software*, processo ou documentação realizada.

Na literatura, de acordo com Pressman (2011), alguns autores utilizam os termos métrica, medida e medição de forma intercambiável e indiscriminadamente. Mas segundo ele, existem diferenças sutis entre os termos:

- Métrica: é a relação entre medidas individuais;
- Medida: fornece uma indicação quantitativa da extensão, quantidade, dimensão, capacidade ou tamanho de algum atributo de um produto ou processo;
- Medição: é o ato de determinar uma medida.

Outro conceito fundamental frequentemente utilizado é o de indicador. Um indicador, segundo Pressman (2011, p. 539), “é uma métrica ou combinação de métricas que proporcionam informações sobre o processo de em um projeto de ou no próprio produto”. O indicador permite ao engenheiro de *software* realizar ajustes no processo, projeto ou produto.

É importante ressaltar que a ISO/IEC 25000 (ISO, 2014), ao contrário da ISO/IEC 9126 (ISO, 1991), não traz a definição do termo métrica. Foi verificado que a norma sugere algumas alterações nas definições dos termos, como: i) medição “*measurement*”, é o conjunto de operações que objetiva determinar o valor de uma medida; ii) medir (*measure)* verbo, é o ato de fazer uma medição; e iii) medida (*measure*) substantivo, é uma variável a qual um valor é atribuído como o resultado de uma medição. Neste trabalho, foram adotadas as definições da norma ISO/IEC 25000 (ISO, 2014), atualmente em vigor.

2.3 Categorias das Medidas de *Software*

Existem várias classificações para as medidas de *software*. O fato de existirem medidas para todo o ciclo de vida do *software* fez com que vários autores como Fenton e Pfleeger (1997), Kan (2002), Pressman (2011) e Sommerville (2011), propusessem suas classificações.

Para os autores Fenton e Pfleeger (1997), Kan (2002) e Pressman (2011), as medidas de *software* podem ser divididas em três categorias: medidas de produto, processo e projeto.

As medidas de produto, segundo Kan (2002), descrevem as características do produto, tais como: tamanho, complexidade, características de *design*, desempenho e nível de qualidade.

As medidas de processo, conforme Pressman (2011), fornecem indicadores que permitem a uma empresa avaliar o processo empregado. Segundo Kan (2002), as medidas de processo podem ser usadas para melhorar o desenvolvimento e a manutenção do *software*. Geralmente, são coletadas durante longos períodos no decorrer dos projetos.

As medidas de projeto, segundo Kan (2002), descrevem as características de um projeto e de sua execução. Além disso, permitem avaliar o andamento de um projeto, custo, quantidade de desenvolvedores, cronograma e produtividade. Essas medidas são denominadas por Fenton e Pfleeger (1997) como medidas de recursos.

De acordo com Sommerville (2011), as medidas podem ser classificadas em medidas de controle e de predição. As medidas de controle, geralmente, são associadas com os processos de *software*. Por exemplo: esforço médio e o tempo necessário para reparar os defeitos reportados. As medidas de predição estão associadas com o produto de *software*. Por exemplo: a complexidade de um módulo, o número de atributos e operações associadas, dentre outros.

A norma ISO/IEC 9126 (ISO, 1991) classifica as medidas em: medidas internas, medidas externas e medidas de qualidade em uso. As medidas internas podem ser aplicadas a um produto de *software* não executável, durante o projeto e a codificação; as medidas externas podem ser utilizadas, para avaliar o comportamento do *software* em ambientes controlados, mediante teste e observação do *software* executável ou do sistema; as medidas de qualidade em uso medem o quanto um produto atende as necessidades dos usuários em um contexto de uso especifico.

2.4 Medidas de Interesse

Na literatura, é encontrada uma ampla variedade de medidas de *software*, dentre elas: as medidas relacionadas à OO (Orientadas a Objetos), descritas na Seção 2.4.1; as medidas de tamanho, descritas na Seção 2.4.2; e medidas de acoplamento, descritas na Seção 2.4.3.

2.4.1 Medidas Relacionadas à Orientação a Objetos

Dois conjuntos de medidas são amplamente referenciados na literatura: as medidas de Chidamber e Kemerer (CK) (CHIDAMBER; KEMERER, 1994); e as medidas *Metrics for Object Oriented Design*, também, conhecida por MOOD (ABREU; CARAPUÇA, 1994). Há, ainda, as medidas da suíte LK, propostas por Lorenz e Kidd (1994) e as medidas da suíte QMOOD, proposta por Bansiya e Davis (2002). As medidas das suítes CK são detalhadas a seguir por serem de interesse deste trabalho.

2.4.1.1 Medidas CK (Chidamber & Kemerer)

Chidamber e Kemerer (1991) apresentaram seis medidas para projetos Orientados a Objetos (OO). Em 1994, os autores apresentaram um estudo complementar ao de 1991. Neste estudo, as medidas propostas foram detalhadas e orientações de como utilizá-las foram apresentadas. Tais medidas ficaram conhecidas como suíte Chidamber & Kemerer ou simplesmente suíte CK.

A suite CK é constituída pelas seguintes medidas:

a) **Métodos Ponderados por Classe (*Weighted Methods per Class – WMC)*:** fornece a complexidade ou tamanho de uma classe calculando, individualmente, a soma da complexidade de cada método de uma classe ou efetuando a contagem dos métodos da classe. O número de métodos e sua complexidade são indicadores razoáveis de quantidade de esforço necessário para desenvolver e testar uma classe (PRESSMAN, 2011). A medida WMC, considerando o cálculo da Complexidade Ciclomática, pode ser expressa como:

$$WMC = \sum_{i=1}^{n} M_i$$

Em que:

n = número total de métodos de uma classe;

Mi = complexidade de cada método.

A medida WMC, considerando a contagem dos métodos, pode ser expressa como:

$$WMC = \sum_{i=1}^{n} 1$$

Em que:

n = número total de métodos de uma classe.

b) **Profundidade de Árvore de Herança (*Depth of Inheritance Tree - DIT):*** é uma medida que fornece a distância máxima de uma classe da árvore de herança até o nó raiz da árvore. Quanto maior a profundidade na árvore de herança, maior a dificuldade de prever o comportamento das classes herdeiras, consequentemente, maior a complexidade do sistema (CHIDAMBER; KEMERER, 1994).

c) **Número de filhos (*Number of children - NOC*):** fornece número imediato de subclasses subordinadas a uma classe. Esta medida calcula o grau de reutilização de uma classe (CHIDAMBER; KEMERER, 1994).

d) **Acoplamento entre Classes de Objetos (*Coupling between object classes - CBO*):** fornece uma medida de acoplamento entre classes. Duas classes estão acopladas quando um método declarado em uma classe utiliza métodos ou variáveis de outra classe. De acordo com Pressman (2011), é provável que a reusabilidade de uma classe diminua à medida que o acoplamento aumente.

e) **Resposta de Classe (*Response for a Class - RFC*):** conforme Chidamber e Kemerer (1994), RFC é um conjunto de métodos que podem ser executados potencialmente em resposta a uma mensagem recebida por um objeto daquela classe. A medida RFC é diretamente proporcional à complexidade geral do sistema. Ou seja, conforme RFC aumenta, a complexidade geral do sistema aumenta.

f) **Ausência de Coesão em Métodos (*Lack of Cohesion in Methods - LCOM*):** fornece uma medida de ausência de coesão entre métodos de uma classe. LCOM é o número de métodos que tem acesso a um ou mais dos mesmos atributos. Se nenhum método tiver acesso ao mesmo atributo, então LCOM = 0. Conforme Chidamber e Kemerer

(1994), quanto maior a similaridade entre métodos, maior será a coesão. Consequentemente, menor o valor de LCOM.

2.4.2 Medidas de Tamanho

Medidas de tamanho quantificam o tamanho de um *software*. Dentre as várias medidas orientadas a tamanho, destacam-se: Linhas de Código (*Lines of Code* – LOC) e Número de Classes (*Number of Class* – nCL).

2.4.2.1 Linhas de Código (LOC) e Número de Classes (nCL)

A medida Linhas de Código (LOC) é uma das medidas mais antigas para mensurar tamanho, esforço e produtividade no desenvolvimento de *software*. Ela consiste na contagem da quantidade do número de linhas de código de um *software*.

O principal problema dessa medida é a ambiguidade da contagem efetiva (KAN, 2002). Quando o *software* era desenvolvido em Assembly, uma linha correspondia a uma instrução. Dessa forma, a definição LOC era objetiva. Contudo, com a ascensão das linguagens de alto nível, a correspondência de um-para-um foi quebrada, ocasionando variações na contagem LOC.

A medida Número de Classes (nCL) calcula a quantidade total de classes em um sistema de *software*.

2.4.3 Medidas de Acoplamento

Medidas de acoplamento mensuram o acoplamento de um *software*, ou seja, calculam o nível de interdependência entre as classes de um *software*.

Dentre as várias medidas relacionadas a acoplamento, destacam-se: Acoplamento Aferente (*Afferent Coupling – Ca*) e Acoplamento Eferente (*Efferent Coupling – Ce*).

2.4.3.1 Acoplamento Aferente (Ca)

A medida Acoplamento, conforme Martin (1995), Aferente calcula o total de classes externas de um pacote que dependem das classes internas a esse pacote. Essa medida, também, é conhecida como *Fan-in* da classe quando calculada em nível de classe. O valor elevado desta medida indica que haverá grande impacto quando ocorrer uma alteração na classe.

2.4.3.2 Acoplamento Eferente (Ce)

A medida Acoplamento Eferente, de acordo com Martin (1995), calcula o total de classes internas a um que depende das classes externas a esse pacote. Essa medida, também, é conhecida como *Fan-out* da classe, quando calculada em nível de classe. Essa medida, também, pode ser um indicador de complexidade. Quanto maior o Acoplamento Eferente, maior a complexidade do *software*.

2.5 Medidas de Qualidade e Valores de Referência

Para manter um nível aceitável de qualidade, um *software* deve ser monitorado e avaliado durante todo processo de desenvolvimento (PRESSMAN, 2011). O monitoramento e a avaliação, geralmente, ocorrem por meio de informações obtidas por medidas de *software*.

As medidas de *software*, muitas vezes, são utilizadas para verificar se produtos, projetos ou processos possuem qualidade. Conforme Pressman (2011), a qualidade pode ser medida ao longo do processo de Engenharia do *Software* ou depois que o *software* for entregue ao cliente e ao usuário. Ainda, de acordo com Pressman (2011), os fatores que afetam a qualidade de *software* podem ser divididos em dois grupos:

(I) Fatores que podem ser medidos diretamente;
(II) Fatores que podem ser medidos indiretamente.

As medidas diretas são obtidas diretamente por meio de uma única medida como, por exemplo, o número de linhas de código (LOC) produzidas, o tamanho de memória ocupado, velocidade de execução, o número de erros registrados num dado período de tempo.

As medidas indiretas são obtidas por meio de uma combinação de medidas como, por exemplo, funcionalidade, qualidade, complexidade, eficiência, manutenibilidade.

Em ambos os fatores devem ocorrer medições. As avaliações de um *software* permitem verificar se ele atingiu um limiar de qualidade necessário (SOMMERVILLE, 2011), comparando as medidas aos valores de referência.

Valores de referência dividem o espaço do valor de uma medida em regiões e, dependendo da região que o valor da medida for, é possível fazer uma avaliação da unidade medida (LANZA; MARINESCU, 2006). Um valor de referência, geralmente, obedece às escalas descritas na Seção 2.3 deste trabalho.

Outros termos estão associados a valores de referência, como *threshold*, limiar, *range* e intervalos. Entende-se *threshold* ou limiar como um valor limite que não deve ser violado. Quando violado, o *software* avaliado ou uma classe medida deve ser revisado. Enquanto *range* ou intervalo é um conjunto de valores que devem ser alcançados ou define um conjunto de valores classificados

qualitativamente, por exemplo, a classificação poderia ser ruim, regular e bom. Neste trabalho, foi utilizado o temo "Valor de Referência", a menos que o contexto imponha interpretação diferente.

Existem duas formas principais que podem ser utilizadas para definir valores de referência para uma medida de *software*: experiência e análise estatística (LANZA; MARINESCU, 2006).

Na literatura, autores como Coppick e Cheatham (1992), French (1999), Lorenz e Kidd (1994) e Rosenberg, Stapko e Gallo (1999) estabeleceram valores de referência para algumas medidas baseando-se na experiência profissional.

Em Rosenberg, Stapko e Gallo (1999), foram propostos valores de referência para o conjunto de medidas da suíte CK (CHIDAMBER; KEMERER, 1994). Os valores de referência foram obtidos por meio de medições de dados de projetos da NASA. Foram analisados, durante 3 anos, mais de 20,000 classes desenvolvidas nas linguagens JAVA e C++. No estudo realizado, foram apresentados valores de referência predominantes e extremos. Os valores predominantes são considerados desejáveis, que representam a maioria dos projetos. Os valores extremos representam o valor máximo (*threshold*).

Em French (1999), foi proposta uma metodologia para estabelecer valores de referência utilizando média e desvio padrão. Os valores foram definidos por meio da análise de um conjunto de 8 sistemas de *software* desenvolvidos nas linguagens Ada95 e C++.

Em Lorenz e Kidd (1994), foram definidos valores de referência para um conjunto de medidas OO. Para isso, foi utilizado um conjunto de projetos da IBM *(International Business Machines Corporation)* desenvolvidos nas linguagens C++ e Smaltalk. Porém, estudos empíricos não foram realizados para validar os valores de referência identificados.

Em Coppick e Cheatham (1992), buscou-se examinar o uso de medidas de complexidade em sistemas OO. Além disso, foi proposto um limiar

(*threshold)* para a medida de Complexidade Ciclomática em um objeto. O valor de referência foi estabelecido pela experiência e mostrou-se razoável para dados coletados na HP (*Hewlett-Packard)*.

Para Alves, Ypma e Visser (2010), uma vez que os valores dependem da experiência de um profissional, é difícil reproduzir ou generalizar. Além disso, a falta de apoio científico leva a questionamentos de sua validade.

Valores de referência, baseados em análise estatística, são derivados com base na análise dos dados de uma população ou amostra. Na literatura, autores como Alves, Ypma e Visser (2010), Catal, Alan e Balkan (2011), El Emam et al. (2002), Ferreira et al. (2009, 2011), Herbold, Grabowski e Waack (2011), Shatnawi (2010) e Shatnawi et al. (2010) propõem a utilização de técnicas estatísticas para derivar valores de referência para um conjunto de medidas.

Em Alves, Ypma e Visser (2010), é apresentada uma metodologia para derivar valores de referência baseados na medição de dados de um *Benchmark*. A metodologia foi aplicada a um conjunto de 100 sistemas OO de código aberto e proprietários desenvolvidos em diversas organizações. Os sistemas utilizados foram desenvolvidos nas linguagens de programação Java e C#, pertenciam a diversos domínios de aplicação e continham diversos tamanhos. Foram estabelecidos valores de referência para medidas em nível de método e classe.

Em Ferreira et al. (2009, 2011), buscou-se identificar valores de referência para um conjunto de medidas de *software* OO. Para isso, foram realizadas análises na distribuição probabilística que modela cada medida. Foram utilizados 40 sistemas de *software* (26.202 classes), de código aberto, desenvolvidos na linguagem de Java, de 11 domínios de aplicação diferentes. Foram identificados valores que podem ser utilizados como referência para as medidas Fator acoplamento (COF - *Coupling Factor*), Ausência de coesão em métodos (LCOM - *Lack of Cohesion in Methods*), Profundidade da árvore de

herança (DIT - *Depth of Inheritance Tree*), Atributos públicos (*Public fields*), Métodos públicos *(Public methods)* e Conexões aferentes (*Afferent couplings*). Tais medidas abrangem fatores como conectividade, ocultação de informação, tamanho da interface, herança e coesão interna.

2.6 Ferramentas de Medição

A coleta manual de medidas em *software* pode ser uma tarefa árdua, que demanda esforço e tempo. Dessa forma, faz-se necessário o uso de ferramentas que realizem coleta automaticamente. Entende-se ferramenta de coleta de medida como um programa que implementa um conjunto de medidas de *software*. Ferramentas permitem avaliar *software*, de acordo com as entidades desejadas e fornece os valores correspondentes para cada medida (LINCKE; LUNKBERG; LÖWE, 2008).

Atualmente, encontram-se disponíveis na literatura, diversas ferramentas (GOOGLE CODEPRO ANALYTIX, 2012; MCCABE IQ, 2012; POWERSOFTWARE, 2014; TERCEIRO et al., 2014) para coleta automática das medidas. Essa variedade de ferramentas permite ao usuário selecionar a ferramenta mais adequada conforme a necessidade. Por exemplo, dependendo da manipulação, da linguagem, do preço, do suporte, da aparência, dos relatórios, dentre outros.

Uma ferramenta é necessária para coleta automatizada de dados, a fim de reduzir o esforço envolvido na coleta dos dados de medidas praticadas atualmente no mercado.

Após realizar uma pesquisa livre na Internet, foram identificadas 8 ferramentas funcionais que fazem medições automaticamente. Outras ferramentas foram identificadas, porém não foram descritas nesta seção, em função da indisponibilidade ou limitações que impediam seu uso, como

indisponibilidade de licenças e falta de instrução de como executar. As ferramentas identificadas são:

- **CKJM:** CKJM é uma ferramenta gratuita, de código aberto que foi desenvolvida por Spinellis (2005), para ilustrar conceitos relacionados à qualidade do código de *software* OO. O desenvolvimento foi motivado pela ausência de ferramentas de medição eficientes e confiáveis capazes de calcular as medidas OO da suíte CK (CHIDAMBER; KEMERER, 1994).
- **CKJM EXT:** CKJM *Ext* (JURECZKO; SPINELLIS, 2010) é uma extensão da ferramenta CKJM (SPINELLIS, 2005). Conforme Jureczko e Spinellis (2010), essa é a versão estendida do CKJM 1.8 e a principal diferença entre a versão original é a inclusão de novas medidas de *software* calculadas. Atualmente, essa ferramenta encontra-se na versão 2.1 de 2011, calcula as medidas da suíte CK e algumas medidas da suíte QMOOD, dentre outras medidas de qualidade.
- **Connecta:** Connecta foi desenvolvida em 2006 para aplicar um Modelo de Avaliação de Conectividade em Sistemas Orientados por Objetos (MACSOO) (FERREIRA et al., 2009). MACSOO é um modelo de reestruturação de *software* baseado em conectividade cujo objetivo é a redução do custo de manutenção.
- **Analizo:** Analizo é uma ferramenta de medição gratuita, multilinguagem e de código aberto desenvolvida por Terceiro et al. (2010), como ferramenta de apoio para sua tese de Doutorado. O desenvolvimento dessa ferramenta foi motivado pela ausência de ferramentas multilinguagem, confiáveis e atuais.

- **Google CodePro Analytix:** Google CodePro Analytix (2012) é uma ferramenta de teste de *software* Java para desenvolvedores que utilizam o Eclipse e estão preocupados com a melhoria da qualidade de *software* e redução dos custos de desenvolvimentos e horários. CodePro Analytix é uma ferramenta gratuita, que se encontra na versão 7.1.0, disponibilizada pelo *Google Develpment*. Embora seja uma ferramenta gratuita, não possui seu código fonte disponível.

- **Code Analyzer:** CodeAnalyzer (CODEANALYZER, 2012) é uma ferramenta multiplataforma, gratuita que analisa o código fonte de *software* desenvolvido nas linguagens Java, C, C++, HTML e Assembly. CodeAnalyzer foi desenvolvida em 2004, atualizada em 2005 e, desde então, não foi mais atualizada.

- **Krakatau:** Krakatau Professional (POWERSOFTWARE, 2012) é uma ferramenta de medição de código fonte comercial desenvolvida pela *Power*. Atualmente, encontra-se na versão 2.11 atualizada em 2013. Essa ferramenta pode realizar medições de sistemas *software* desenvolvidos nas linguagens C/C++ e Java.

- **Simple Code Metrics:** A ferramenta SimpleCodeMetrics (SIMPLECODEMETRICS, 2012) é um *plug-in* para o IDE (Integrated Development Environment) NetBeans. Essa ferramenta foi desenvolvida em 2008 e, desde então, não foi atualizada. SimpleCodeMetrics não possui documentação, suporte ou informações que permitam detalhar as medidas implementadas. A ausência de uma descrição adequada das medidas dificulta identificar o que realmente está sendo medido.

2.7 Revisão Sistemática da Literatura

De acordo com Biolchini et al. (2005), uma Revisão Sistemática da Literatura (RSL) ou, simplesmente, Revisão Sistemática, é uma técnica baseada em evidências que teve origem na medicina e ciências médicas. Tal técnica vem sendo empregada em diversas áreas, dentre elas, a Engenharia de *Software*. Segundo Mafra, Barcelos e Travassos (2006), o primeiro trabalho que estabeleceu um paralelo entre a medicina e a Engenharia de *Software* foi proposto em 2004 por Kitchenham (2004).

Uma RSL consiste em uma técnica empregada para identificar, interpretar e avaliar pesquisas relevantes disponíveis para uma questão de pesquisa, temática ou fenômeno de interesse (KITCHENHAM, 2004).

Segundo Randolph (2009), uma RSL pode ser conduzida para delimitar um problema de pesquisa, buscar novas linhas de pesquisa, evitar pesquisas infrutíferas, dentre outros. A realização de uma RSL é de suma importância a qualquer projeto de pesquisa (DYBA; DINGSOYR; HASSEN, 2007).

Ao contrário de uma revisão narrativa tradicional, uma RSL é um resumo conciso das melhores evidências disponíveis (BIOLCHINI et al., 2005). Em uma RSL, utilizam-se métodos rigorosos para identificar, avaliar criticamente e sintetizar artigos científicos relevantes sobre um determinado tema. Esses métodos são definidos com antecedência e são documentados para que outros possam apreciar criticamente (DYBA; DINGSOYR; HASSEN, 2007).

Uma RSL é caracterizada como um estudo secundário, pois visa reunir as melhores evidências científicas da literatura para responder questões relevantes. Os estudos individuais que contribuem para uma RSL são chamados de estudos primários, por exemplo: estudos empíricos, estudo de caso e *surveys*.

Geralmente, estudos secundários são realizados para agregar valor a um estudo primário.

2.7.1 Processo de construção de uma Revisão Sistemática.

Na literatura, podem ser encontradas diferentes sugestões sobre o número e a ordem das atividades realizadas em uma RSL. Conforme Biolchini et al. (2005) e Kitchenham e Charters (2007), a condução de uma RSL pode ser dividida em três fases (Figura 1): i) planejamento; ii) execução e iii) análise dos resultados.

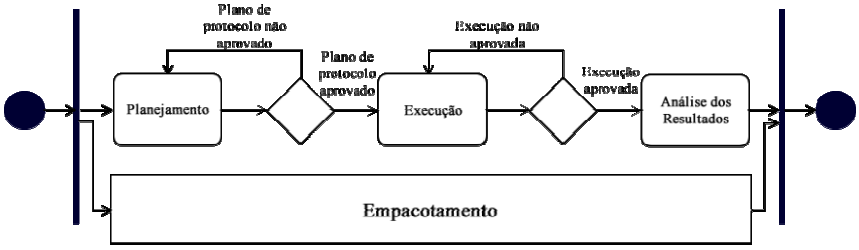

Figura 1 Fases da uma RSL.
Fonte: Adaptado de Biolchini et al. (2005).

Na fase de planejamento, os objetivos da pesquisa são descritos e respostas às questões iniciais são obtidas, como as que guiarão os estudos, sendo respondidas ao final da RSL. Nessa fase, descrevem-se os objetivos e as motivações que levaram à realização da pesquisa. Elaboram-se questões referentes ao que se deseja encontrar nos resultados da RSL. E desenvolve-se um protocolo em que são formuladas questões relacionadas ao que se pretende alcançar com a RSL e; revisa-se, cuidadosamente, o protocolo desenvolvido.

Na fase de execução, ocorre uma investigação nas fontes definidas, segundo critérios de inclusão e exclusão definidos no protocolo, para realizar o estudo e a classificação dos artigos encontrados. É necessário verificar os resultados obtidos com as buscas e repetir a investigação, o estudo e a classificação. Nessa fase, são realizadas buscas nas fontes definidas no protocolo e filtragens das evidências encontradas; em seguida, os resultados são sumarizados conforme critérios previamente estabelecidos. O processo de execução é iterativo, ou seja, a busca pode ser reajustada e executada novamente caso os resultados não tenham sido razoáveis.

Na fase de análise dos resultados, realiza-se a coleta, a organização e a análise dos dados extraídos dos artigos científicos. Esses dados são analisados e sintetizados conforme o método escolhido. Ao final, os resultados são analisados de maneira global, gerando melhor planejamento, caso necessário.

2.7.2 Protocolo

O protocolo é um elemento crítico, pois é utilizado para nortear a execução da RSL. Nele, são descritos os critérios a serem seguidos na avaliação dos artigos. A sua elaboração ocorre na fase de planejamento da RSL e exige conhecimento prévio do assunto em questão, para que os tópicos descritos selecionem, realmente, apenas os artigos científicos relevantes para a pesquisa.

A existência de um protocolo predefinido reduz a influência do pesquisador. Existem vários modelos e *templates* de protocolos na literatura. Neste trabalho, foi adotado o modelo proposto por Biolchini et al. (2005).

Conforme Biolchini et al. (2005), a estrutura básica de um protocolo, geralmente, obedece aos seguintes tópicos:

1) Objetivos: define-se o que se pretende alcançar com os resultados ao término da aplicação da RSL;
2) Questão de Pesquisa: elaboram-se perguntas a serem respondidas com o estudo;
3) Palavras-Chave: formulam-se palavras-chave referentes à questão de pesquisa. Para elaborar as palavras-chave, necessita-se de conhecimento prévio do tema abordado;
4) *Strings* de Busca: cria-se *string* utilizada para encontrar artigos científicos nas fontes selecionadas. As palavras-chave geralmente são combinadas por meio de operadores booleanos, como: AND, OR ou NOT;
5) Método de Busca de Fontes: define-se onde serão encontrados os artigos científicos;
6) Listagem de Fontes: restringem-se os locais onde a busca serão realizadas;
7) Tipos de estudo primários: definem-se os tipos dos estudos que se buscam nos artigos científicos;
8) Idioma dos estudos primários: define-se o idioma a ser considerado;
9) Critérios de Inclusão e Exclusão dos Artigos: elaboram-se critérios para descarte dos artigos científicos.

3 METODOLOGIA

Neste Capitulo, são apresentadas as atividades realizadas durante a pesquisa.

3.1 Atividades

As principais atividades realizadas neste trabalho são apresentadas na Figura 2.

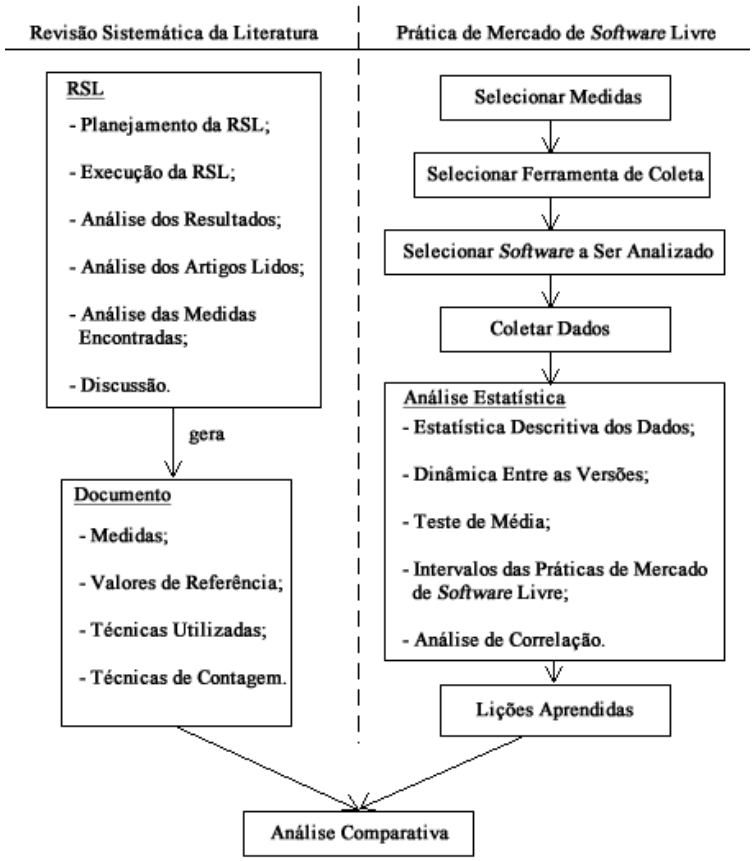

Figura 2 Etapas para consecução dos objetivos deste trabalho.

Observa-se, na Figura 2, que as atividades ou principais ações foram divididas em 2 grandes grupos denominados “Revisão Sistemática da Literatura” e “Prática de Mercado de *Software* Livre”. Em Revisão Sistemática da Literatura, a principal atividade está relacionada ao planejamento e execução de uma RSL, a fim de identificar o histórico e estado da arte na literatura acadêmica. Em Prática de Mercado de *Software* Livre, as principais atividades estão relacionadas com o processo de coleta e análise de dados junto ao mercado de *software* livre.

As próximas seções descrevem em mais detalhes cada um dos grandes grupos e as principais ações contidas em cada um deles.

3.1.1 Revisão Sistemática da Literatura

Nesta etapa, aplica-se a RSL passando por cada uma das atividades descritas em RSL na Figura 2.

Em “Planejamento da RSL”, aplica-se um protocolo bem definido para se planejar a RSL, evitando definições durante sua execução que seja apenas de conveniência do aplicador. Em “Execução da RSL”, executa-se o planejamento realizado a priori. Em “Análise dos Resultados”, apresenta-se um resumo critico dos artigos identificados após a execução da RSL. Em “Análise dos Artigos Lidos”, analisa-se cada um dos artigos lidos, a fim de obter uma visão geral dos artigos identificados. Em “Análise de Medidas Encontradas”, analisam-se algumas medidas de *software* e os valores de referência identificados na RSL. Em “Discussão”, discutem-se os principais resultados obtidos com a RSL. Após aplicar a RSL, gera-se um documento, contendo as medidas e seus valores de referência, as técnicas utilizadas para determinar os valores de referência e as técnicas de contagem. Entende-se, como técnica de contagem, as diferentes formas de calcular uma medida. Por exemplo, considerar linhas de comentários

na medida LOC seria uma forma de calcular. Não considerar linhas de comentários, seria outra forma.

3.1.2 Prática de Mercado de *Software* Livre

Descrevem-se, nesta etapa, as atividades relacionadas às práticas de mercado de *software* livre, realizadas ao longo deste trabalho.

3.1.2.1 Selecionar Medidas

Nesta etapa, define-se um conjunto de medidas a serem utilizadas para comparar as práticas do mercado e o preconizado na literatura científica. Esta atividade é influenciada e influencia a atividade de Selecionar Ferramenta de Coleta, pois é temporalmente inviável capturar medidas no mercado sem uma ferramenta automatizada.

3.1.2.2 Selecionar Ferramenta de Coleta

Nesta etapa, define-se uma ou mais ferramentas a serem utilizadas na coleta das medidas selecionadas. Esta atividade é influenciada e influencia a atividade de Selecionar Medidas, pois é inviável selecionar ferramentas que capturem medidas de pouco ou nenhum interesse/impacto para o mercado e/ou academia.

3.1.2.3 Selecionar *Software* a ser Analisado

Nesta etapa, realiza-se o levantamento de um conjunto de *software* a ser analisado. Esta atividade é influenciada pela atividade de Selecionar Ferramenta

de Coleta, pois os sistemas de *software* a serem analisados devem ser compatíveis com a ferramenta de coleta selecionada.

3.1.2.4 Coletar Dados

Nesta etapa, realiza-se a coleta dos dados, aplicando-se a(as) ferramenta(s) selecionada(s) na etapa de "Selecionar Ferramenta de Coleta" aos sistemas de *software* a ser analisados, selecionados na etapa "Selecionar *Software* a ser Analisado".

3.1.2.5 Análises Estatísticas

Nesta etapa, realizam-se análises estatísticas nos dados coletados, passando por cada uma das atividades descritas em Análises Estatísticas na Figura 2.

Em "Estatística Descritiva dos Dados", resumem-se os dados coletados por meio de estatísticas descritivas, como: média, mediana, moda, desvio padrão, variância, assimetria, curtose, valor mínimo e valor máximo. Em "Dinâmica Entre as Versões", realiza-se um estudo para verificar a dinâmica "evolução" entre as versões dos dados coletados. Em "Em Teste de Média", realizam-se testes de média para verificar a existência de diferenças significativas entre versões de um mesmo *software* e aplica-se o teste *post-hoc* para comparar todos os sistemas de *software* entre si. Em "*Intervalos das Práticas de Mercado de Software Livre*", obtêm-se, para as medidas selecionadas, valores de referência dos valores praticados no mercado de *software* livre. Em "Análise de Correlação", realiza-se uma análise de correlação entre todos os pares das medidas selecionadas.

3.1.2.6 Lições Aprendidas

Nesta etapa, apresentam-se as lições aprendidas ao longo do processo de obtenção dos valores praticados no mercado.

3.1.3 Análise Comparativa

Nesta etapa, realiza-se uma análise comparativa entre os valores de referência identificados na literatura com os valores praticados no mercado.

4 REVISÃO SISTEMÁTICA DA LITERATURA

Neste Capítulo, são apresentados e discutidos os resultados encontrados com a Revisão Sistemática da Literatura, ao longo da pesquisa. Apresenta-se a descrição do protocolo, seguindo os moldes definidos em Biolchini et al. (2005) e aplicação do protocolo ao tema. Em seguida, apresentam-se as fases de execução e a análise dos resultados.

4.1 Planejamento da RSL

- **Objetivos:** O objetivo desta RSL foi de realizar um levantamento de artigos científicos que apresentam medidas de *software* que possuem valores de referência associados a ela.
- **Questões de Pesquisa:** quais medidas de *software* possuem valores de referência atribuídos a elas? Quais são os valores de referência identificados na literatura?
- **Palavras-Chave:** as palavras-chave foram definidas, por meio de reuniões com os envolvidos na pesquisa e mediante leitura de artigos relevantes relacionados ao tema, sendo escolhidas as mais comuns. Também foram utilizados sinônimos e termos de mesmo propósito. As palavras-chave adotadas foram: *software, metric, measure, measuring, thresholds, reference values, values, range, limits.*
- ***String* de Busca:** a *string* de busca foi elaborada com base nas palavras chave, relacionando-as logicamente.

  (software) AND (metric OR metrics OR measure OR measures OR measuring) AND ("reference value" OR "reference values" OR ranges OR thresholds OR limits OR range OR threshold OR limit).

- **Método de Busca de Fontes:** sites de bibliotecas científicas virtuais.
- **Listagem das Fontes:**
  - *IEEE Xplore (http://ieeexplore.ieee.org);*
  - *Elsevier Science Direct (http://www.sciencedirect.com);*
  - *Scopus (http://www.scopus.com);*
  - *ACM Library (http://dl.acm.org);*
  - *Ei Compendex (http://www.engineeringvillage2.org).*
- **Tipos de Artigos:** foram considerados artigos completos sobre medidas de *software*, comparações e análises.
- **Idioma dos Artigos:** acredita-se que os principais artigos científicos relevantes à pesquisa estejam em inglês e português. Dessa forma, os artigos devem estar em inglês ou português.
- **Critérios de inclusão ou exclusão dos artigos:** os artigos devem: i) Estar disponíveis na web; ii) Estar em inglês ou português; iii) Apresentar valores de referência para medidas de *software*; iv) Ter sido publicado entre os anos de 1990 e 2012; e v) Oferecer conteúdo completo.

4.2 Execução da RSL

A fase de execução de uma RSL deve ser rigorosa e capaz de encontrar todos os artigos científicos relacionados ao tema (KITCHENHAM, 2004). Conforme descrito na Seção 2.7.1, deste trabalho, a *fase de Execução* subdivide-se em quatro etapas, que são: *Obtenção da Pesquisa; Seleção Primária; Seleção Secundária; e Organização dos Resultados,* apresentadas nas Seções seguintes.

4.2.1 Etapa 1 - Obtenção da Pesquisa

Na Obtenção da pesquisa, foram realizadas buscas nas bases: *Ei Compendex, IEEE Xplore, ACM Library, Elsevier Science Direct e Scopus*. Durante as buscas, filtragens foram realizadas, como: filtragem do ano (entre o período de 1990 e 2012); filtragem do idioma (inglês e português); filtragem da área (Ciência da Computação e/ou Engenharia de *Software*). Nessa etapa, os artigos científicos foram pesquisados nas bases de dados com base nos *Títulos* e *Palavras-Chave*.

Quadro 1 *Strings* de busca utilizada na etapa 1

Base de dados	***String* de busca**	
IEEE Xplore	*(() AND (metric OR metrics OR measure OR measures OR measuring) AND ("reference value" OR "reference values" OR ranges OR thresholds OR limits OR range OR threshold OR limit))*	
Elsevier Science Direct	*pub-date > 1989 and TITLE-ABS-KEY(() AND (metric OR metrics OR measure OR measures OR measuring) AND ("reference value" OR "reference values" OR ranges OR thresholds OR limits OR range OR threshold OR limit))[All Sources(Computer Science)]*	
Ei Compendex	*(((((() AND (metric OR metrics OR measure OR measures OR measuring) AND ("reference value" OR "reference values" OR ranges OR thresholds OR limits OR range OR threshold OR limit)) WN KY) AND (({computer } OR { engineering}) WN CV)) AND ((({english} OR {portuguese}) WN LA) AND (1990-2012) WN YR))*	
Scopus	*TITLE-ABS-KEY(() AND (metric OR metrics OR measure OR measures OR measuring) AND ("reference value" OR "reference values" OR ranges OR thresholds OR limits OR range OR threshold OR limit)) AND PUBYEAR > 1989 AND (LIMIT-TO(LANGUAGE, "English")) AND (LIMIT-TO(SUBJAREA, "COMP"))*	
Elsevier Science Direct	*pub-date > 1989 and TITLE-ABS-KEY(() AND (metric OR metrics OR measure OR measures OR measuring) AND ("reference value" OR "reference values" OR ranges OR thresholds OR limits OR range OR threshold OR limit))[All Sources(Computer Science)]*	
ACM Library	*String 1*	*(“ measure*”) AND (“reference value*” OR range* OR threshold* OR limit*)*
	String 2	*(“ measuring”) AND (“reference value*” OR range* OR threshold* OR limit*)*
	String 3	*(“ metric*”) AND (“reference value*” OR range* OR threshold* OR limit*)*

Por causa das características de algumas máquinas de busca, a *string* de busca definida no protocolo sofreu leves alterações (sem perder sua representatividade). Em algumas situações, foi necessário incluir parâmetros à *string* de busca. Na *ACM Library*, foi necessário dividir a *string* de busca em três para obter um resultado plausível de análise. As buscas foram realizadas entre os dias 02 e 03 de maio de 2012. As *strings* de busca, utilizadas nessa RSL, são apresentadas no Quadro 1.

As execuções foram criteriosamente aplicadas por dois pesquisadores e os artigos científicos foram selecionados com base nos critérios de Inclusão e Exclusão estabelecidos no protocolo. Os artigos que não atenderam aos critérios de inclusão definidos no protocolo foram descartados. Nessa etapa, a ferramenta JabRef 2.7.2 (JABREF, 2012) foi utilizada para listar e manipular as referências dos artigos científicos. Ao término dessa etapa, foram encontrados 4.357 artigos científicos (Tabela 1) baseados na *string* de busca definida no protocolo.

4.2.2 Etapa 2 - Seleção Primária

Depois de realizada a seleção inicial (Etapa 1), os artigos científicos foram submetidos a uma seleção primária em que os *Títulos*, *Resumos* e *Palavras-Chave* foram analisados e filtrados manualmente. Durante a filtragem, verificou-se que grande quantidade de artigos científicos pertencia a outras áreas da computação e não se enquadravam aos propósitos dessa RSL. Com isso, o número de artigos científicos passou de 4.357 (obtidos na seleção inicial) para 184.

Em uma RSL, as buscas iniciais retornam grande quantidade de estudos irrelevantes que não respondem às questões ou não têm relação com o tema em questão (KITCHENHAM; CHARTERS, 2007).

4.2.3 Etapa 3 - Seleção Secundária

Na seleção secundária, os 184 artigos científicos selecionados na seleção primária (Etapa 2) passaram por uma inspeção onde se fez a leitura dos *Resumos, Introduções e Conclusões*. Nessa etapa, também, foram verificados se os artigos eram irrelevantes, repetidos ou incompletos.

Dos 184 artigos selecionados: 137 foram considerados irrelevantes, visto que não correspondiam ao objetivo da RSL; 28 artigos foram considerados repetidos, ou seja, foram encontrados em mais de uma base de dados; nenhum artigo foi considerado incompleto, todos os artigos pesquisados estavam disponíveis. Por fim, 19 artigos científicos passaram pelos critérios de seleção. A Tabela 1 apresenta a sumarização dos resultados da execução.

Tabela 1 Sumarização dos resultados da RSL.

Bases	**Busca Inicial**	**Seleção Primária**	**Seleção Secundária**			**Incluídos**
			Irrelevantes	**Repetidos**	**Incompletos**	
IEEE Xplore	1.803	84	73	0	0	11
Elsevier	121	14	11	0	0	3
Compendex	1.086	21	7	11	0	3
Scopus	1.107	36	21	14	0	1
ACM Library	240	29	25	3	0	1
Total	**4.357**	**184**	**137**	**28**	**0**	**19**

4.2.4 Etapa 4 – Organização dos Resultados

Verifica-se que a base que retornou o maior número de artigos relevantes foi o *IEEE Xplore* com 58% dos estudos. As bases que apresentaram o menor número de estudos relevantes foram: *Scopus e ACM Library* com 5% dos artigos científicos.

O Quadro 2 apresenta os artigos científicos relevantes que respondem as questões de pesquisa estabelecidas no protocolo dessa RSL. São apresentados o ano de publicação dos artigos, o título dos trabalhos, a referência e a base de dados onde foram encontrados.

Quadro 2 Artigos incluídos identificados com a RSL

Índice	Ano	Titulo	Referência	Base
A	2009	An outlier detection algorithm based on object-oriented metrics thresholds	Alan e Catal (2009)	IEEE
B	2010	Deriving metric thresholds from benchmark data	Alves, Ypma e Visser (2010)	IEEE
C	2011	Benchmark-Based Aggregation of Metrics to Ratings	Alves, Correia e Visser (2011)	IEEE
D	2000	Thresholds for object-oriented measures	Benlarbi et al. (2000)	IEEE
E	2002	The optimal class size for object-oriented *software*	El Emam et al. (2002)	IEEE
F	2009	Clustering and Metrics Thresholds Based *Software* Fault Prediction of Unlabeled Program Modules	Catal, Sevim e Diri (2009)	IEEE
G	2003	A metrics suite for measuring reusability of *software* components	Washizaki, Yamamoto e Fukazawa (2003)	IEEE
H	2007	Observing Distributions in Size Metrics: Experience from Analyzing Large *Software* Systems	Ramler, Wolfmaier e Natschlager (2007)	IEEE
I	1997	*Software* metrics model for quality control	Schneidewind (1997)	IEEE
J	2010	A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems	Shatnawi (2010)	IEEE
K	2009	Reference Values for Object-Oriented *Software* Metrics	Ferreira et al. (2009)	IEEE
L	2011	Identifying thresholds for object-oriented *software* metrics	Ferreira et al. (2011)	Elsevier
M	2011	Class noise detection based on *software* metrics and ROC curves	Catal, Alan e Balkan (2011)	Elsevier
N	2011	Improving the applicability of object-oriented class cohesion metrics	Dallal (2011)	Elsevier
O	2010	Finding s*oftware* metrics threshold values using ROC curves	Shatnawi et al. (2010)	Compendex
P	1992	*Software* metrics for object-oriented systems	Coppick e Cheatham (1992)	Compendex
Q	2005	An empirical exploration of the distributions of the Chidamber and Kemerer object-oriented metrics suite	Succi et al. (2005)	Compendex
R	2011	Calculation and optimization of thresholds for sets of *software* metrics	Herbold, Grabowski e Waack (2011)	Scopus
S	2010	Estimation of *Software* Reusability: An Engineering Approach	Nair e Selvarani (2010)	ACM

4.3 Análise dos Resultados da RSL

Apresentam-se, nesta seção, a análise e a discussão dos 19 artigos científicos identificados que respondem a questão de pesquisa estabelecida no protocolo dessa RSL. Além disso, são apresentados as medidas e os seus valores de referência e algumas técnicas utilizadas para definir valores de referência e de contagem.

A) An outlier detection algorithm based on object-oriented metrics thresholds (ALAN; CATAL, 2009).

Em Alan e Catal (2009), é proposto um algoritmo de detecção de *outlier* que se baseia em *threshold* de seis medidas de *software* da suíte CK (CHIDAMBER; KEMERER, 1994). Tais *thresholds* foram obtidos com base na documentação da ferramenta McCabe IQ (2012), que afirma que os *thresholds* foram obtidos de acordo com padrões da indústria e experiência profissional.

Também é interessante notar que McCabe (MCCABE SOFTWARE, 2012) define "*threshold*" como o valor de uma medida acima que um elemento é de interesse. Normalmente, abaixo de um limite, as medidas não se correlacionam com efeitos reais e elementos de código abaixo do limiar, geralmente, não necessitam de modificações ou revisões no código. Ou seja, os *thresholds* apresentados são valores que provocam preocupação quando ultrapassados. Nenhuma consideração é feita para os limiares inferiores, o que poderia desencadear preocupação se o valor medido for muito baixo.

Na Tabela 2 apresentam-se as medidas e os seus *thresholds*. A medida LOCM definida em McCabe IQ (2012) fornece a ausência de coesão, como LCOM (com O e C invertido), mas a sua definição não é claramente a mesma que outras disponíveis na literatura.

Tabela 2 *Thresholds* recomendados em *McCabe Tool* (MCCABE IQ, 2012) e referenciados em Alan e Catal (2009).

Medidas	***Thresholds***
Weighted Methods per Class (WMC)	14
Depth of Inheritance Tree (DIT)	7
Number of Children (NOC)	3
Coupling Between Object Classes (CBO)	2
Response for a Class (RFC)	100
Lack of Cohesion of Methods (LOCM)	75

***B) Deriving metric thresholds from benchmark data* (ALVES; YPMA; VISSER, 2010).**

Em Alves, Ypma e Visser (2010), é apresentado um método para derivar empiricamente *thresholds* de medidas de *software* segundo medição de dados de um *benchmark*. O método foi aplicado a um *benchmark* com 100 sistemas OO, tanto proprietários quanto de código aberto, de diferentes domínios de aplicação. A técnica utilizada foi a análise da distribuição probabilística das medidas. Os autores afirmam que os *thresholds* identificados são generalizáveis.

As medidas utilizadas, bem como os valores sugeridos, são exibidas na Tabela 3. Os percentis apresentados na Tabela 3 são utilizados para caracterizar o código, de acordo com quatro categorias: risco baixo (entre 0 - 70%); risco moderado (70 - 80%); risco alto (80 - 90%) e risco muito alto (> 90%). Também é possível observar, na Tabela 3, que a medida *Fan-in* tem baixo risco até o valor de 29, envolvendo 70% do código. A coluna seguinte, que representa o código *Fan-in*, de 29-42 é de risco moderado e assim por diante. Não foi explicado o termo "*per file*". Não se observa se todos os sistemas possuem uma classe por arquivo ou cada arquivo pode suportar mais de uma classe. Infelizmente, esta interpretação interfere nos resultados.

Os *thresholds*, apresentados na Tabela 3, foram derivados segundo a porcentagem geral que deseja representar. Por exemplo: para a medida de Complexidade Ciclomática, 90% do código global, o valor derivado foi 14.

Tabela 3 *Thresholds* sugeridos em Alves, Ypma e Visser (2010).

Medidas	***Thresholds***		
	70%	80%	90%
Cyclomatic Complexity (per method)	6	8	14
Lines of Code (per Method)	30	44	74
Fan-in (per file)	10	22	56
Number of Methods (per file)	29	42	73
Medida	80%	90%	95%
Number of Parameters (per method)	2	3	4

Tal como acontece com Alan e Catal (2009), discutido na seção anterior, apenas *thresholds* para valores altos são considerados. A novidade neste trabalho é a existência de *thresholds* diferentes, indicando níveis de preocupação crescente.

C) Benchmark-Based Aggregation of Metrics to Ratings (ALVES; CORREIA; VISSER, 2011).

Em Alves, Correia e Visser (2011), é apresentada uma metodologia baseada em uma abordagem de classificação em duas etapas, em que (i) medidas são comparados aos *thresholds* para resumi-los em perfis de risco, e (ii) os perfis de risco são mapeados para classificações.

Essa metodologia complementa o trabalho anterior de Alves, Ypma e Visser (2010). Eles agregam medições individuais em uma escala de classificação n-ponto com base em *thresholds*. Das medidas apresentadas na Tabela 4, apenas a medida *Number of Methods* não foi incluída neste estudo.

Tabela 4 Valores da medida *Fan-in* (em nível de arquivo) identificados em Alves, Correia e Visser (2011).

Intervalos	**Baixo Risco [0; 10]**	**Risco Moderado [10; 22[**	**Risco Alto [22; 56[**	**Risco Muito Alto [56; ∞[**
5	-	23.9	12.8	6.4
4	-	31.2	20.3	9.3
3	-	34.5	22.5	11.9
2	-	41.8	30.6	19.6

D) Thresholds for object-oriented measures (BENLARBI et al., 2000).

Em Benlarbi et al. (2000), foi utilizada uma técnica estatística para avaliar a hipótese de que existem efeitos de *thresholds* em *software*. Tais efeitos são baseados na teoria cognitiva que afirma que acima dos *thresholds* haveria aumento rápido na dificuldade de compreender *software*, e, portanto, as taxas de erro aumentariam rapidamente.

O estudo foi baseado em dois sistemas de telecomunicações desenvolvidos em C++ e 4 medidas da suíte CK (CHIDAMBER; KEMERER, 1994). Foi descoberto que não houve efeito de *thresholds*. Isso não significa, no entanto, que não é possível desenvolver *thresholds*, será discutido posteriormente.

Benlarbi et al. (2000) apresentam quatro *thresholds* encontrados na literatura, tal como mostrado na Tabela 5.

Tabela 5 *Thresholds* citados em Benlarbi et al. (2000).

Medidas	***Thresholds***	**Ref.**
Weighted Methods per Class (WMC)	100	Rosenberg, Stapko e Gallo (1999)
Coupling Between Object Classes (CBO)	5	
Response for Class (RFC)	100	
Depth of Inheritance Tree (DIT)	6	Lorenz e Kidd (1994)

E) The optimal class size for object-oriented (EL EMAM et al., 2002).

Em El Emam et al. (2002), os autores avaliam a conjectura *Goldilocks* em sistemas OO. Esta conjectura é representada por uma curva em forma de “U”, na relação entre o tamanho e falhas. Quatro medidas de tamanho foram avaliadas por meio da medição de três sistemas comerciais desenvolvidas em C++ e Java. Pelo estudo mostra-se que não há evidência empírica para a Conjectura *Goldilocks*, em outras palavras, não existe um tamanho claramente melhor.

Foram calculados *thresholds*, com base na análise estatística, em um contexto específico. Os *thresholds* são apresentados na Tabela 6. Os limites sugeridos tendem a ser pouco razoável. Por exemplo, os *thresholds* sugerem que uma classe não deve ter mais do que um método. Esses limites não devem, portanto, ser tomados como orientação, mas para ilustrar como os autores enfatizam que sua técnica matemática poderia identificar qualquer *thresholds* razoável. A principal conclusão deste estudo foi que, embora a propensão de falha aumente com o tamanho do sistema, não há ponto de corte real (*threshold*) acima do qual há aumento repentino na propensão a falhas.

Tabela 6 *Thresholds* estimados em El Emam et al. (2002).

Medidas	***Thresholds***
Number of Statemenst (STMTS)	Menor que 671
Source Lines of Code *(*SLOC)	Acima de 3
Number of Methods (NM)	Acima de 1
Number of Attributes (NAI)	Menor que 39

F) Clustering and Metrics Thresholds Based Fault Prediction of Unlabeled Program Modules (CATAL; SEVIM; DIRI, 2009).

Em Catal, Sevim e Diri (2009), os autores identificam módulos propensos a falhas que não possuem histórico de falhas, por meio de técnicas de clusterização e *thresholds* de medidas de *software*. Foram utilizadas seis medidas, dentre elas, as medidas de McCabe (1976) e Halstead (1977). Os *thresholds* utilizados (Tabela 7) foram propostos por *Integrated Metrics, Inc*. (ISM), mas nenhuma documentação referente aos valores foi encontrada.

Foi verificado que a medida LOC é o número total de linhas de código, incluindo linhas em branco e linhas de comentários, mas os autores não deixam claro se a granularidade para LOC foi por método, classe ou arquivo. As medidas e *thresholds* utilizados em Catal, Sevim e Diri (2009) são apresentadas na Tabela 7.

Tabela 7 *Thresholds* citados por Catal Sevim e Diri (2009).

Medidas	*Thresholds*
Lines of Code (LOC)	65
Cyclomatic Complexity (CC)	10
Unique Operator Count (UOp)	25
Unique Operand Count (UOpnd)	40
Total Operator Count (TOp)	125
Total Operand Count (TOpnd)	70

G) A metrics suite for measuring reusability of components (WASHIZAKI; YAMAMOTO; FUKAZAWA, 2003)

Em Washizaki, Yamamoto e Fukazawa (2003), os autores propõem cinco medidas para mensurar a capacidade de reutilização de componentes, com base em informações limitadas, que podem ser obtidas sem qualquer código fonte. As cinco medidas calculam compreensibilidade, adaptabilidade e portabilidade de um componente. Os *thresholds* foram definidos com base na análise estatística de componentes *JavaBeans*. Os valores podem ser observados na Tabela 8.

Tabela 8 Valores de referência propostos em Washizaki, Yamamoto e Fukazawa (2003).

Medidas	*Thresholds*
Existence of Meta-Information	[0.5, 1.0]
Rate of Component Observability	[0.17, 0.42]
Rate of Component Customizability	[0.17, 0.34]
Self-Completeness of Component's Return Value	[0.61, 1.0]
Self-Completeness of Component's Parameter	[0.42, 0.77]

H) Observing Distributions in Size Metrics: Experience from Analyzing Large Systems (RAMLER; WOLFMAIER; NATSCHLAGER, 2007).

Em Ramler, Wolfmaier e Natschlager (2007), os autores comparam as distribuições dos valores de medidas de tamanho mediante a análise, utilizando diagramas de Pareto, de diferentes sistemas de *software*, bem como versões consecutivas de um único sistema. Os *thresholds* utilizados neste estudo foram

obtidos baseados em Wolfmaier e Ramler (2005). Segundo Ramler, Wolfmaier e Natschlager (2007), os valores pertencem a um contexto geral e foram propostos com base na experiência profissional. No entanto, não foram encontradas evidências para validar os valores apresentados. As medidas e seus limiares são apresentados na Tabela 9.

Tabela 9 *Thresholds* utilizados em Ramler, Wolfmaier e Natschlager (2007).

Medidas	***Thresholds***
Number of Public Method per Class	5 a 10
Number of Public Attribute per Class	0
Lines of Codes	5 até 1.000
Lines of Comment (LoCo) per File (número de '//' e '/* */'. Blocos de comentários são contados como n linhas)	20% das linhas são comentários. 10% dos arquivos possuem > 40 LoCo 30% dos arquivos possuem [10;39] LoCo 60% dos arquivos possuem < 10 LoCo
Class Outbound Call Class	0 a 20

I) Metrics model for quality control (SCHNEIDEWIND, 1997)

Em Schneidewind (1997), foi apresentado um modelo das relações entre fator de qualidade e medidas de qualidade. O primeiro representa a qualidade na visão do cliente e usuários, enquanto o segundo é facilmente mensurável pelos desenvolvedores. Foi utilizada estatística não paramétrica para identificar *thresholds* para 13 medidas de *software*, apresentados na Tabela 10, no contexto do programa *Space Shuttle*. Em seguida, os *thresholds* foram validados empiricamente. As medidas foram classificadas, de acordo com o conceito do domínio, o que significa que as medidas de classificação superior mapeiam melhor fatores de qualidade. A Tabela 10 é ordenada de acordo com esta análise.

Também foi utilizada a análise marginal, para determinar quantas medidas devem ser coletadas. Além de um determinado valor, a informação extra não é útil.

Os *thresholds* foram importantes em Schneidewind (1997) para determinar a aceitação ou rejeição de módulos de *software* para Aeronaves *Space Shuttle*.

Tabela 10 *Thresholds* identificados em Schneidewind (1997).

Medidas	***Thresholds***
Total comment count	38
Unique operator count	10
Total statement count (no comments)	26
Total node count (in control graph)	11
Total edge count (in control graph)	10
Total non-commented lines *of code*	19
Maximum path length (edges in control graph)	8
Total path count (in control graph)	1
Average path length (edges in control graph)	4
Unique operand count	33
Total operator count	26
Total operand count	21
Total cycle count (in control graph)	0

J) A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems (SHATNAWI, 2010)

Em Shatnawi (2010), foram utilizados modelos estatísticos, derivados da regressão logística, para identificar valores de referência para três medidas da suíte CK (CHIDAMBER; KEMERER, 1994). A metodologia foi validada empiricamente em três versões do projeto eclipse. Os valores de referência são apresentados na Tabela 11.

Tabela 11 Valores de referência apresentados em Shatnawi (2010).

Medidas	*Thresholds* Rosenberg (1998)	*Thresholds* Shatnawi (2010)
Weighted Methods per Class (WMC)	100	20
Coupling Between Object Classes (CBO)	5	9
Response for Class (RFC)	100	40

K) Reference Values for Object-Oriented Metrics (FERREIRA et al., 2009).

Em Ferreira et al. (2009), os autores propõem valores de referência para um conjunto de medidas de *software* OO com base na análise das distribuições probabilísticas de um grande volume de *software* de *opensource*, desenvolvido na linguagem Java. As medidas utilizadas abrangem fatores, como: conectividade, ocultação de informação, tamanho da interface, herança e coesão interna. Os valores de referência (Tabela 12) são generalizáveis e podem ser aplicados em diferentes domínios de aplicação.

Tabela 12 Valores de referência apresentados em Ferreira et al. (2009).

Medidas	**Valores de Referência**		
	Bom	**Regular**	**Ruim**
Coupling Factor (COF)	até 0,02	Entre 0,02 e 0,14	>= 0,14
Lack of Cohesion in Methods (LCOM)	0	Entre 0 e 20	>= 20
Depth of Inheritance Tree (DIT)		Valor típico 2	
# Afferent Couplings	1	Entre 1 e 20	>= 20
# Public Fields	0	Entre 0 e 8	>= 8
# Public Methods	0 a 10	Entre 10 e 40	>= 40

L) *Identifying thresholds for object-oriented metrics* (FERREIRA et al., 2011).

Em Ferreira et al. (2011), os autores analisaram sistemas adicionais e refinaram o seu trabalho baseados em Ferreira et al. (2009), propondo valores ligeiramente adaptados para os valores de referência. Esses podem ser observados na Tabela 13. Os valores foram obtidos por meio da análise das distribuições probabilísticas de medidas, considerando diferentes domínios de aplicação, tipos e tamanhos.

Tabela 13 *Thresholds* propostos em Ferreira et al. (2011).

Medidas	***Thresholds***		
	Bom	**Regular**	**Ruim**
Coupling Factor (COF)	Até 0,02	]0,02; 0,14]	> 0,14
Lack of Cohesion in Methods (LCOM)	0	[1; 20]	> 20
Depth of Inheritance Tree (DIT)		Valor típico 2	
# Afferent Couplings	1	]2; 20]	> 20
# Public Fields	0	]1; 10]	> 10
# Public Methods	[0;10]	[11; 40]	> 40

***M) Class noise detection based on metrics and ROC curves* (CATAL; ALAN; BALKAN, 2011).**

Em Catal, Alan e Balkan (2011), os autores apresentam uma abordagem de detecção de ruído com base em *thresholds* de medidas de *software*. Os *thresholds* foram obtidos tomando por base a análise *Receiver Operating Characteristic* (ROC), em um estudo de caso de cinco projetos da NASA. As quatro primeiras linhas, apresentadas na Tabela 14, são medidas de McCabe (MCCABE, 1976). As outras linhas são as medidas propostas por Halstead (1977).

Thresholds distintos foram identificados para cada um dos projetos, esses são mostrados na Tabela 14. Infelizmente, os autores não deixam claro como algumas medidas são calculadas, por exemplo, na medida *Lines of Code*, não se verifica se linhas em branco, linhas de comentários, comentários de uma linha ou comentários de *n* linhas foram consideradas.

Tabela 14 *Thresholds* definidos em Catal, Alan e Balkan (2011).

Medidas	Projeto 1	Projeto 2	Projeto 3	Projeto 4	Projeto 5
Lines of Code	11	32	33	28	22
Cyclomatic complexity	3	5	5	3	4
Essential complexity	2	4	4	2	4
Design complexity	3	3	3	3	3
Line count	5	24	27	2	22
Count of lines of comments	1	1	4	7	4
Count of blank lines	1	6	4	7	1
Count of lines of code and comment	1	1	1	0	7
Unique operator	7	12	15	18	15
Unique operand	7	17	19	21	20
Total operator	13	42	54	53	50
Total operand	8	27	36	57	34
Branch count	4	4	8	5	7

N) Improving the applicability of object-oriented class cohesion metrics **(DALLAL, 2011).**

Em Dallal (2011), o autor objetiva melhorar a aplicabilidade de uma variedade de medidas de coesão, ajustando suas fórmulas, considerando casos especiais, como: classes sem métodos, classes sem atributos, dentre outros. Em seguida, foram apresentados os valores médios e o terceiro quartil das medidas melhoradas, com base em medições de quatro sistemas. Essas são mostradas na Tabela 15. As medidas melhoradas foram validadas empiricamente, de acordo com a sua capacidade para prever falhas, mas isso não significa que a validação dos valores foi realizada adequadamente.

Tabela 15 Valores de referência definidos em Dallal (2011).

Medidas	**Valores de Referência**	
	Média	**75%**
LLD Similarity-based Class Cohesion (LSCC)	0.32	0.56
Lack of Cohesion of methods (LCOM1)	42.05	21
Lack of Cohesion of methods (LCOM2)	26.88	8
Tight Class Cohesion (TCC)	0.5	1
Loose Class Cohesion (LCC)	0.58	1
Degree of Cohesion (DCd)	0.5	1
Degree of Cohesion (DCi)	0.58	1
Class Cohesion (CC)	0.35	0.6
Sensitive Class Cohesion Metric (SCOM)	0.39	0.86
Lack of Cohesion of methods (LCOM3)	1.67	2
Lack of Cohesion of methods (LCOM4)	1.62	2
Lack of Cohesion of methods (LCOM5)	0.76	1
Coh	0.43	0.75
Cohesion Among Methods in a Class (CAMC)	0.42	0.62
Normalized Hamming Distance (NHD)	0.46	0.72

O) *Finding metrics threshold values using ROC curves* (SHATNAWI et al., 2010).

Em Shatnawi et al. (2010), foram identificados *thresholds* para medidas de *software* OO, aplicando curvas ROC (*Receiver Operating Characteristic*), a fim de separar classes de alto impacto. Foram estabelecidos *thresholds* para um conjunto de medidas da suíte CK (CHIDAMBER; KEMERER, 1994), considerando três versões do projeto Eclipse. Os valores foram validados empiricamente, mas se trata de um estudo preliminar. Os autores apresentam os *thresholds* estabelecidos em Rosenberg (1998) e os valores obtidos utilizando curvas ROC (Tabela 16).

Tabela 16 *Thresholds* definidos em Shatnawi et al. (2010).

Medidas	***Thresholds* Rosenberg (1998)**	***Thresholds* Shatnawi et al. (2010)**
Weighted Methods per Class (WMC)	100	24
Coupling Between Object Classes (CBO)	5	13
Response for Class (RFC)	100	44
Number of Operations (NOO)	-	9
Coupling through message passing (CTM)	-	33

***P) Metrics for object-oriented systems* (COPPICK; CHEATHAM, 1992).**

Em Coppick e Cheatham (1992), os autores examinam a aplicação de medidas de complexidade de *software* dentro do paradigma OO e sugerem um *threshold* para a medida Complexidade Ciclomática de uma classe, com base na experiência profissional. Os autores afirmam que os valores de referência propostos foram razoáveis para projetos da *Hewlett-Packard* (HP), mesmo que nenhuma validação empírica tenha sido realizada. Os limiares são apresentados na Tabela 17.

Tabela 17 *Thresholds* apresentados em Coppick e Cheatham (1992).

Medidas	***Thresholds***
Cyclomatic Complexity (method and function)	10
Cyclomatic Complexity (Class)	100

Q) An empirical exploration of the distributions of the Chidamber and Kemerer object-oriented metrics suite (SUCCI et al., 2005).

Em Succi et al. (2005), os autores apresentam uma investigação meta-analítica de medidas OO de CK (CHIDAMBER; KEMERER, 1994) em dois aspectos: a presença de colinearidade entre as medidas CBO, NOM e RFC e as faixas estreitas de NOC e DIT. Os *thresholds* foram obtidos pela análise estatística de mais de 200 projetos de código aberto, mas também citam outros valores propostos em Chidamber e Kemerer (1994). Os valores citados e propostos são apresentados na Tabela 18.

Tabela 18 *Thresholds* apresentados em Succi et al. (2005).

Medidas	**Valores de Referência Chidamber e Kemerer (1994)**	***Thresholds* Succi et al. (2005)**	
		JAVA	**C++**
Depth of Inheritance Tree (DIT)	10	< 6	< 6
Number of Children (NOC)	10	> 4 e < 6	< 6

R) Calculation and optimization of thresholds for sets of metrics (HERBOLD; GRABOWSKI; WAACK, 2011).

Em Herbold, Grabowski e Waack (2011), os autores apresentam um método algorítmico para calcular *thresholds* de um conjunto de medidas, aplicando aprendizado de máquina e técnicas de mineração de dados. Na terceira coluna da Tabela 19 apresentam-se *thresholds* utilizados como base nas análises e valores adaptados considerando trabalhos anteriores (BENLARBI et al., 2000; COPELAND, 2005; FRENCH, 1999; LORENZ; KIDD, 1994) ou determinados com base na experiência dos autores.

Foram realizados quatro estudos de caso: dois avaliam a metodologia proposta e outros dois reproduzem a classificação. Os estudos de caso abrangem *software* de diferentes linguagens e analisam em nível de classe e método.

Os *thresholds* não foram validados, os autores concentraram na metodologia proposta. Um ponto chave neste trabalho são os autores calcularem *thresholds*, para serem usados em conjunto, com base na análise de métodos ou classes que tiveram problemas. Outro ponto chave é a abordagem permitir que um desenvolvedor estabeleça seus próprios limites, para projeto de *software*, com base em seu histórico de manutenção.

Na Tabela 19 mostram-se os *thresholds* da linha de base e os valores calculados pelo método proposto nos seus estudos de caso. É possível observar que os *thresholds* calculados e da linha de base foram os mesmos, sugerindo que os valores iniciais podem ser generalizados.

Tabela 19 *Thresholds* citados em Herbold, Grabowski e Waack (2011).

Medidas	**Ling.**	***Thresholds* citados**	**Ref.**	***Thresholds* calculados**
Medidas de métodos e funções				
Cyclomatic Number (VG)	C	24	French (1999)	24
	C++	10		10
	C#	10		9
Nested Block Depth (NBD)	C	5		5
	C++	5		5
	C#	5		5
N. of Function Calls (NFC)	C	5	-	5
	C++	5	-	5
	C#	5	-	5
N. of Statements (NST)	C	50	-	50
	C++	50	-	50
	C#	50	-	50
Medidas de Classe				
Weighted Methods per Class (WMC)	Java	100		99
Coupling Between Object Classes (CBO)	Java	5	Benlarbi et al. (2000)	5
Response for Class (RFC)	Java	100		98
N. of Overridden Methods (NORM)	Java	3	Lorenz e Kidd (1994)	3
Lines of Code (LOC)	Java	500	Copeland (2005)	500
N. of Methods (NOM)	Java	20		20
N. of Static Methods (NSM)	Java	4	Lorenz e Kidd (1994)	4

***S) Estimation of Reusability: An Engineering Approach* (NAIR; SELVARANI, 2010).**

Em Nair e Selvarani (2010), os autores propõem um modelo de estimativa eficaz para a reutilização de *software*. O modelo prevê a reutilização com base em *thresholds* de três medidas da suíte CK (CHIDAMBER; KEMERER, 1994): DIT, NOC e WMC.

Os *thresholds* foram propostos em Rosenberg e Hyatt (2001), com base em dados da NASA, por meio da experiência profissional. Os limiares citados são apresentados na Tabela 20.

Os autores apresentaram curvas para identificar a influência das medidas DIT, RFC e WMC na reutilização. Por exemplo, quando DIT aumenta de 1 a 6, os aumentos de reutilização variam de 9% a 98%, aproximadamente.

Tabela 20 *Thresholds* citados em Nair e Selvarani (2010).

Medidas	***Thresholds* Rosenberg e Hyatt (2001)**	
	Desejável	**Max**
Weighted Methods per Class (WMC)	20	100
Depth of Inheritance Tree (DIT)	3	6
Response for Class (RFC)	[50, 100]	222

4.4 Análise dos Artigos Lidos

Mediante os artigos científicos obtidos na RSL, verifica-se que, nos últimos anos, o número de artigos relacionados a valores de referência de medidas de *software* tem aumentado. Um dos fatores que contribuem para esse aumento é a exigência do mercado por produtos de qualidade. Verifica-se que 57,8% dos artigos científicos encontrados nessa RSL foram publicados desde 2009, como apresentado na Figura 3.

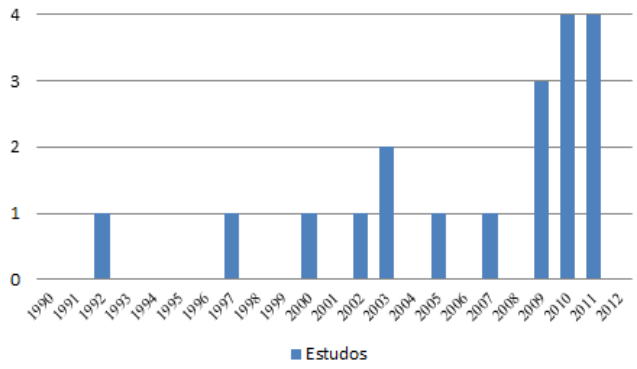

Figura 3 Quantidade de artigos distribuídos por ano.

Considerando os 19 artigos encontrados, foram identificadas 65 medidas que apresentam valores de referência. Dentre elas, foram encontradas medidas típicas do paradigma OO e medidas tradicionais que foram adaptadas para o paradigma OO, como as medidas LOC e Complexidade Ciclomática. No total, 57.4% dos artigos encontrados referenciam as medidas típicas do paradigma OO, visto que, 82.5% deles referenciam medidas da suíte CK (CHIDAMBER; KEMERER, 1994).

O Quadro 3 sumariza os artigos analisados. Na coluna um, encontra-se o índice dos artigos analisados; na coluna dois, os artigos são classificados em: i) Tipo I - estudos que utilizam valores de referência existente para atingir um objetivo, como: detectar *outlier* e prever falha; e ii) Tipo II - estudos que objetivam estabelecer e/ou otimizar valores de referência para medidas de *software*.

Os valores de referência utilizados nos estudos do Tipo I foram obtidos por meio de ferramentas ou por meio de outros estudos. Os valores obtidos por ferramentas, como *McCabe IQ, ISM ou Sotograph,* não fornecem informações que permitam detalhá-las, pois sua documentação não está prontamente disponível. Entre os artigos identificados, 31,6% são do Tipo I e 68,4% são do Tipo II, como mostrado na Tabela 21.

Quadro 3 Outros resultados dos artigos identificados na RSL.

Índice	Classificação	Ref. a outros artigos ou ferramentas	Validação Empírica	Técnica	Contexto
A	Tipo I	McCabe IQ (2012)	Não	Experiência	Especifico
B	Tipo II	McCabe (1976)	Não	Análise das Distribuições	Genérico
C	Tipo I	Alves, Ypma e Visser (2010)	Não	Análises Estatísticas	Especifico
D	Tipo I	Lorenz e Kidd (1994) e Rosenberg, Stapko e Gallo (1999)	Negativo	Experiência	Especifico
E	Tipo II	Benlarbi et al. (2000)	Negativo	Análises Estatísticas	Especifico
F	Tipo I	ISM*	Não	Regressão Logística	Especifico
G	Tipo II	-	Sim	Análises Estatísticas	Especifico
H	Tipo I	Wolfmaier e Ramler (2005)	Não	Experiência	Genérico
I	Tipo II	-	Sim	Análises Estatísticas	Especifico
J	Tipo II	Rosenberg, Stapko e Gallo (1999)	Sim	Análises Estatísticas	Especifico
K	Tipo II	-	Não	Análise das Distribuições	Especifico
L	Tipo II	Ferreira et al. (2009)	Sim	Análises Estatísticas	Especifico
M	Tipo II	-	Não	Curvas ROC	Especifico
N	Tipo II	-	Não	Experiência	Genérico
O	Tipo II	Rosenberg (1998)	Não	Curvas ROC	Especifico
P	Tipo II		Não	Experiência	Especifico
Q	Tipo II	Rosenberg (1998)	Não	Análises Estatísticas	Genérico
R	Tipo I	Benlarbi et al. (2000), Copeland (2005), French (1999) e Lorenz e Kidd (1994),	Não	Aprendizado de Máquina	Especifico
S	Tipo II	Rosenberg (1998)	Não	Experiência	Especifico

ISM*- *Integrated Metrics, Inc. URL http://www.ismwv.com* indisponível.

Os rótulos "não", "sim" e "negativo", mostrados no Quadro 3 e resumidos na Tabela 22 significam, respectivamente, que "não houve validação", "houve validação", ou "houve validação, mas os resultados indicam que os valores de referência são ruins". Os artigos de Benlarbi et al. (2000) e El Emam et al. (2002) tiveram validação negativa, o que representa 10,5% dos artigos, como mostrado na Tabela 22. Também é possível observar que 21,1% dos artigos validam os valores de referência, e em 68,4% não houve validação. Estes

resultados são indesejáveis, pois somente 21,1% dos valores propostos são validados e, dentre eles, somente o artigo de Ferreira et al. (2011) afirma que os valores podem ser genéricos. A Engenharia de *Software* deve ter valores de referência bem definidos e validados, a fim de apoiar os engenheiros de *software* durante o processo de desenvolvimento e tomada de decisão.

Tabela 21 Porcentagens de artigos classificados quanto ao tipo.

Classificação	Quantidade	%
Tipo I	6	31.6%
Tipo II	13	68.4%
∑	19	100%

Em outros artigos valida-se apenas o método usado para identificar os valores de referência, mas não validam os valores tal como apresentado nos artigos de Herbold, Grabowski e Waack (2011) e Nair e Selvarani (2010).

Tabela 22 Percentual dos artigos que apresentaram validações.

Valores Validados	Quantidade	%
Sim	4	21.1%
Não	13	68.4%
Negativa	2	10.5%
∑	19	100%

Quanto às técnicas utilizadas para a obtenção dos valores de referência, segundo Lanza e Marinescu (2006), há duas abordagens principais: experiência profissional e análise estatística. Dos artigos analisados, 31,6% obtiveram seus valores pela experiência profissional e 68,4% obtiveram pela análise estatística, como apresentado na Tabela 23. Os outros métodos destacados na coluna cinco do Quadro 3, como aprendizado de máquina e curvas ROC, foram considerados análise estatística.

Tabela 23 Porcentagens de artigos que utilizaram experiência ou estatística.

Classificação	Quantidade	%
Experiência profissional	6	31.6%
Análises estatísticas	13	68.4%
∑	19	100%

O contexto foi classificado como genérico e específico. O rótulo “genérica” indica os valores de referência que satisfazem os seguintes critérios: a) três ou mais sistemas; b) mais do que 50 % de sistemas são desenvolvidos por pessoas diferentes; c) mais de um domínio de aplicação; e d) mais de uma linguagem de programação. Caso contrário, “específico” é usado no rótulo. Nos artigos analisados, como apresentado na Tabela 24, 79% foram classificados em um contexto específico, e 21% parecem ter aplicabilidade geral.

Tabela 24 Porcentagens de artigos específicos ou de aplicabilidade geral.

Classificação	**Quantidade**	**%**
Aplicabilidade Especifica	15	79%
Aplicabilidade Genérica	4	21%
∑	19	100%

Durante o processo de análise, vários métodos foram encontrados para calcular os valores de referência. Esses incluem a experiência, a análise estatística, modelos de erro, *clustering*, análise de distribuição e aprendizagem de máquina.

Foi observado que a maioria dos artigos não é replicável em razão de dados incompletos, tais como falta de versões e nome dos sistemas utilizados, descrição detalhada das medidas utilizadas e detalhes sobre as ferramentas utilizadas. Esses dados devem ser incluídos nos artigos para maior compreensão dos valores obtidos. Ainda, seria desejável estabelecer um protocolo para orientar autores e interessados, sobre quais informações precisam ser incluídas nos artigos que permitem a replicação e validação.

Outro problema encontrado foram artigos que não definem instruções de como cada medida, realmente, está sendo calculada. Por exemplo, qual é a diferença entre as medidas "comentários" e "linhas de comentários"? Como foram contados “linhas de comentário”? Por fim, outra dificuldade foi determinar se um valor refere-se a um mínimo ou máximo, por exemplo, no artigo de Schneidewind (1997).

No artigo de Shatnawi et al. (2010), os autores utilizaram três versões do Eclipse para determinar valores. No entanto, o uso de diferentes versões do mesmo *software* não resultará no mesmo nível de generalidade, é como se sistemas completamente diferentes fossem usados. O mesmo argumento pode ser feito quando são analisados vários sistemas no mesmo domínio.

Na Tabela 57, presente no Apêndice A, é apresentado um detalhamento das informações dos valores de referência identificados na literatura, como domínios de aplicação, *software* usados para definir os valores de referência e porte dos sistemas de *software* utilizados. No Quadro 4, presente no Apêndice B, são apresentadas diferentes definições teóricas das medidas identificadas na literatura.

4.5 Análise das Medidas Encontradas

Nesta seção, são analisados os valores de referência identificados com os artigos encontrados na Revisão Sistemática da Literatura.

4.5.1 *Weight Methods per Class (WMC)*

Duas interpretações foram encontradas para a medida WMC: a primeira interpretação, chamada aqui de WMC1, é calculada simplesmente contando o número de métodos em uma classe. A segunda interpretação, chamada aqui de WMC2, é calculada pela soma da Complexidade Ciclomática de McCabe (1976) dos métodos da classe.

Na Tabela 25, é possível observar que apenas um valor foi encontrado para WMC1 e valores, significativamente, diferentes foram encontrados para WMC2. Pela mesma interpretação, verifica-se que o *threshold* pode ser de 20 ou 100 como ilustrados, respectivamente, em Shatnawi (2010). Neste caso, é difícil ou impossível determinar qual valor deve ser considerado, pois a variação foi significativa.

Tabela 25 Valores de medida WMC.

WMC				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
WMC1 (Contagem dos Métodos)	**A**	Alan e Catal (2009)	14	Max
WMC2 (Soma das Complexidades Ciclomática)	**D**	Benlarbi et al. (2000)	100	Max
	J	Shatnawi (2010)	100	Max
	J	Shatnawi (2010)	20	Max
	O	Shatnawi et al. (2010)	24	Max
	O	Shatnawi et al. (2010)	100	Max
	R	Herbold, Grabowski e Waack (2011)	100	Max
	S	Nair e Selvarani (2010)	20 e 100	Desejável e Max

4.5.2 *Depth of Inheritance Tree (DIT)*

Definições consistentes foram encontradas para a medida DIT. Esta medida calcula a profundidade máxima da árvore de herança em um sistema. Na Tabela 26, observa-se que os valores de 6 a 10 são, frequentemente, encontrados como sendo o valor máximo sugerido. Entretanto, em virtude da variação dos valores máximos, mais investigações seriam necessárias para apontar se um sistema seria pior com DIT de 6 ou 10.

Tabela 26 Valores da medida DIT.

DIT			
Índice	**Ref.**	**Valor**	**Natureza da Medida**
A	Alan e Catal (2009)	7	Max
D	Benlarbi et al. (2000)	6	Max
K	Ferreira et al. (2009)	2	Típico
L	Ferreira et al. (2011)	2	Típico
Q	Succi et al. (2005)	10	Max
Q	Succi et al. (2005)	6 (em Java e C++)	Max
S	Nair e Selvarani (2010)	3 e 6	Desejável e Max

4.5.3 *Number of Children (NOC)*

A medida NOC calcula o número de filhos imediato que uma determinada classe tem. Na Tabela 27, verifica-se que existem valores diferentes para o valor máximo ou limite superior, que varia de 3 a 10. Assim como as

medidas anteriores, existe uma variação considerável entre os valores máximos encontrados e mais investigação deve ser realizada a fim de definir um valor máximo para a medida NOC.

Tabela 27 Valores da medida NOC.

NOC			
Índice	**Ref.**	**Valor**	**Natureza da Medida**
A	Alan e Catal (2009)	3	Max
Q	Succi et al. (2005)	10	Max
Q	Succi et al. (2005)	]4, 6[Java e <6 C++	Desejável e Max

4.5.4 *Lack of Cohesion Metric (LCOM)*

A medida LCOM calcula a ausência de coesão e possui diversas interpretações e nomes diferentes, como mostrado na Tabela 28. Ao longo do tempo, diversas interpretações e visões diferentes de coesão surgiram. Com isso, novas medidas foram propostas objetivando apoiar essas interpretações. Neste trabalho, não serão explicadas as variações da medida LCOM. Tais explicações podem ser encontradas em Dallal (2011).

Tabela 28 Valores das medidas LCOM 1, 2, 3, 4, 5 e LOCM.

LCOM				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
LCOM1	**N**	Dallal (2011)	42 e 21	Media e 75th percentil
LCOM2	**K**	Ferreira et al. (2009)	0,]0;20[e >=20	Bom, Regular e Ruim
	L	Ferreira et al. (2011)	0, [1;20] e >20	Bom, Regular e Ruim
	N	Dallal (2011)	27 e 8	Media e 75th percentil
LCOM3	**N**	Dallal (2011)	1.67 e 2	Media e 75th percentil
LCOM4	**N**	Dallal (2011)	1.62 e 2	Media e 75th percentil
LCOM5	**N**	Dallal (2011)	0.76 e 1	Media e 75th percentil
LOCM (McCabe Tools)	**A**	Alan e Catal (2009)	75	Max

4.5.5 *Response For Class (RFC)*

A medida RFC corresponde ao número de métodos que podem ser executados em resposta a uma mensagem recebida por um objeto da classe. A medida é dada pela quantidade de métodos da classe somada à quantidade de métodos invocados por cada método da classe.

Na Tabela 29 são apresentados os valores máximos identificados para a medida RFC. Verifica-se que os valores variam de 40 a 222. Entretanto, mais investigações seriam necessárias, pois a variação foi considerável e não permite apontar um valor que possa ser utilizado com base nos valores propostos.

Tabela 29 Valores da medida RFC.

RFC			
Índice	**Ref.**	**Valor**	**Natureza da Medida**
A	Alan e Catal (2009)	100	Max
D	Benlarbi et al. (2000)	100	Max
J	Shatnawi (2010)	100	Max
J	Shatnawi (2010)	40	Max
O	Shatnawi et al. (2010)	100	Max
O	Shatnawi et al. (2010)	44	Max
R	Herbold, Grabowski e Waack (2011)	100	Max
R	Herbold, Grabowski e Waack (2011)	98	Max
S	Nair e Selvarani (2010)	[50;100] e 222	Desejável e Max

4.5.6 *Coupling Factor (COF)*

A medida COF é dada pela razão entre o número total de conexões existentes entre as classes do *software* e o maior número possível de conexões para o *software*.

Na Tabela 30, são apresentados os valores identificados para a medida COF. Tais valores foram classificados em bom, regular e ruim. Entretanto, os

valores identificados em K e L são oriundos de um mesmo estudo e sugerem que COF <= 0.02 são valores bons, e acima de 0.14, são valores ruins.

Tabela 30 Valores da medida COF.

COF			
Índice	**Ref.**	**Valor**	**Natureza da Medida**
K	Ferreira et al. (2009)	<= 0.02,]0.02;0.14[e >= 0.14	Bom, Regular e Ruim
L	Ferreira et al. (2011)	<= 0.02,]0.02; 0.14] e > 0.14	Bom, Regular e Ruim

4.5.7 *Coupling Between Object Classes (CBO)*

A medida CBO contabiliza o número de classes às quais uma determinada classe está acoplada. Existe acoplamento entre duas classes quando métodos de uma delas usam métodos ou variáveis de outra (CHIDAMBER; KEMERER, 1994).

Na Tabela 31 é possível observar valores de referência diferentes, para a medida CBO, em cada estudo. Na primeira linha, da Tabela 31, o valor máximo de CBO é 2 e na linha 8, é apresentado o valor máximo 13. Sendo os dois valores máximos, fica impossível dizer qual é o valor máximo adequado em decorrência da grande variação dos valores apresentados.

Tabela 31 Valores da medida CBO.

CBO			
Índice	**Ref.**	Valor	Natureza da Medida
A	Alan e Catal (2009)	2	Max
D	Benlarbi et al. (2000)	5	Max
J	Shatnawi (2010)	5	Max
J	Shatnawi (2010)	9	Max
O	Shatnawi et al. (2010)	5	Max
O	Shatnawi et al. (2010)	13	Max
R	Herbold, Grabowski e Waack (2011)	5	Max

4.5.8 *Afferent Coupling (Ca), Fan-in*

A medida Ca fornece o número total de classes externas de um pacote que depende de classes de dentro desse pacote. Quando calculada no nível da classe, essa medida, também, é conhecida como *Fan-in* da classe.

Na Tabela 32 apresentam-se os valores identificados para as medidas Ca e *Fan-in*. Observa-se que os valores propostos para a medida Ca são de um mesmo estudo, embora existam leves alterações de um intervalo para o outro. Como os valores da medida Ca foram sugeridos com base na medição de classes, então, essa medida pode ser comparada a *Fan-in*, pois por definição são semelhantes.

Verifica-se a grande variação dos valores sugeridos. Para a medida *Fan-in*, foi sugerido o valor 73, referente ao percentil 90% e, para a medida Ca, valores maiores ou iguais a 20. Com base nesses valores, não é possível identificar um intervalo que possa ser utilizado como referência, para isso, seria necessário realizar mais estudos para identificar o valor máximo adequado.

Tabela 32 Valores das medidas Ca e *FAN-IN*.

Ca e FAN-IN				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
FAN-IN	**B**	Alves, Ypma e Visser (2010)	10,22 e 56	70%, 80%, 90% percentis
AC	**K**	Ferreira et al. (2009)	0,]1;20[e >=20	Bom, Regular e Ruim
	L	Ferreira et al. (2011)	1, [2;20] e >20	Bom, Regular e Ruim

4.5.9 *Cyclomatic Complexity (CC)*

A medida Complexidade Ciclomática corresponde ao número de caminhos linearmente independentes em fluxo do programa e tem,

significativamente, diferentes valores de referência nos artigos estudados, como mostrado na Tabela 33.

Tabela 33 Valores das medidas de CC.

CC				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
Per Method	**B**	Alves, Ypma e Visser (2010)	<=6]6;8]]8;14] >14	Risco Baixo Risco Moderado Risco Alto Risco muito Alto
	F	Catal, Sevim e Diri (2009)	10	Impossível verificá-lo em virtude da indisponibilidade da referência original.
	M	Catal, Alan e Balkan (2011)	P1 - 3 P2 - 5 P3 - 5 P4 - 3 P5 - 4	Melhores valores para cada projeto P1, P2,…, P5
	P	Coppick e Cheatham (1992)	10	Max
	R	Herbold, Grabowski e Waack (2011)	C - 24 C++ - 10 C# - 10	Max
Per Module	**P**	Coppick e Cheatham (1992)	100	Max (considerando 10 métodos e cada um suportando complexidade máxima igual a 10)
Design Complexity Per Module (Number of paths including calls to other modules)	**M**	Catal, Alan e Balkan (2011)	P1 - 3 P2 - 3 P3 - 3 P4 - 3 P5 - 3	Melhores valores para cada projeto P1, P2,…, P5

4.5.10 *Operator and Operand Countings*

As medidas de Halstead calculam a quantidade de Operandos Únicos, Operadores Únicos, Total de Operadores Únicos e Total de Operandos Únicos.

Verifica-se, analisando a Tabela 34, que existem diferentes valores de referência para cada conjunto de dados da NASA, em Catal, Alan e Balkan (2011) denominados P1, P2, ..., P5. As medidas de Halstead, chamadas de medidas diretas, são utilizadas para calcular medidas indiretas, por exemplo, o volume de *software* pode ser utilizado para indicar a complexidade. Quanto maior for o volume de *software*, maior será a sua complexidade.

Tabela 34 Valores das medidas de contagem de Operandos e Operadores.

Índice	**Ref.**	**ÚNICO**		**TOTAL**	
		Operador	**Operando**	**Operador**	**Operando**
F	Catal, Sevim e Diri (2009)	25	0	125	70
I	Schneidewind (1997)	10	33	26	21
M	Catal, Alan e Balkan (2011)	P1 - 7 P2 - 12 P3 - 15 P4 - 18 P5 - 15	P1 - 7 P2 - 17 P3 - 19 P4 - 21 P5 - 20	P1 - 13 P2 - 42 P3 - 54 P4 - 53 P5 - 50	P1 - 8 P2 - 27 P3 - 36 P4 - 57 P5 - 34

4.5.11 *Number of Attributes (NOA), Number of Methods (NOM) e Number of Parameters (NOP*)

Na Tabela 35 são apresentadas medidas relacionadas a classes e métodos. É possível observar medidas que envolvam número de atributos e métodos por classe e o número de parâmetros por método.

Algumas variações são consideradas, como atributos públicos e privados. Na Tabela 35 é possível observar que, no artigo de índice H, foi sugerido o valor máximo de 0 para a medida número atributos público (NOAP), em função de boas práticas sugeridas para modelagem OO. Tais sugestões podem ser consideradas uma recomendação teórica.

No entanto, é possível observar que o artigo de índice K e L, o valor 0 é classificado como bom, mas valores até 10 são considerados regulares. Tais

intervalos foram obtidos considerando a análise da distribuição de dezenas de sistemas de código aberto. Dessa forma, as sugestões podem ser consideradas uma recomendação prática.

Tabela 35 Valores das medidas NOA, NOM, NOP e NOAP.

Número de Atributos, métodos e parâmetros.				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
Número de Atributos	**E**	El Emam et al. (2002)	39	*Threshold* inválido
Número de Atributos Públicos	**K**	Ferreira et al. (2009)	0,]1;8[, >=8	Bom, Regular e Ruim
	L	Ferreira et al. (2011)	0, [1;10], >10	Bom, Regular e Ruim
	H	Ramler, Wolfmaier e Natschlager (2007)	0	Max
Número de Métodos	**B**	Alves, Ypma e Visser (2010)	29,42 e 73	70%, 80% 90% percentil
	E	El Emam et al. (2002)	1	*Threshold* inválido
	R	Herbold, Grabowski e Waack (2011)	20	Max
Número de Métodos Públicos	**K**	Ferreira et al. (2009)	[0;10],]10;40[, >=40	Bom, Regular e Ruim
	L	Ferreira et al. (2011)	[0;10], [11;40], >40	Bom, Regular e Ruim
Número de Parâmetros	**B**	Alves, Ypma e Visser (2010)	2,3 e 4	70%, 80% 90% percentil

4.6 Discussão

Infelizmente, não foi encontrado valor de referência que possa ser utilizado, imediatamente, sem qualquer verificação ou adaptação. Foi constatada

a existência de diferenças significativas entre os valores das medidas apresentadas.

Os artigos de Ferreira et al. (2009, 2011) parecem ter valores de referência confiáveis, considerando a quantidade de sistemas e domínios. Entretanto, os valores de referência foram definidos, considerado apenas sistemas Java e isso restringe a utilização dos valores em sistemas desenvolvidos em outras linguagens de programação.

Em alguns artigos, valores de referência foram propostos, mas nos artigos não se explicam como foram calculados ou a razão para esses valores. Às vezes, afirmou-se que os valores foram estabelecidos com base na experiência profissional (COPPICK; CHEATHAM, 1992; MCCABE IQ, 2012). Neste último caso, deve-se, simplesmente, confiar? A ciência foi feita para ser exata e existe um longo percurso para melhorar as medidas de *software* e sua utilização.

No artigo de Catal, Alan e Balkan (2011), foram apresentados valores diferentes para cinco projetos, como apresentado na Tabela 14. Também foram observadas diferenças significativas entre os valores de determinadas medidas em diferentes artigos. Assim, surge a questão de saber se é possível calcular um único valor de referência geral, aplicável em muitos casos.

Além disso, os artigos de Benlarbi et al. (2000) e El Emam et al. (2002) mostram resultados negativos para essas validações. No primeiro (BENLARBI et al., 2000), os autores demonstraram que não houve efeito limiar em algumas medidas. No segundo (EL EMAM et al., 2002), os autores demonstraram que não há evidência empírica para a conjectura "*Goldilocks*".

Neste contexto, sem valores de referência que podem ser utilizados de forma genérica, trazem-se à mente os processos de aprendizagem, como: aprendizagem de máquina, redes neurais, lógica *fuzzy*, e assim por diante. Esses métodos, que não fornecem valores de referência genéricos, têm que ser treinados e possuem aplicação limitada em cada contexto.

Considerando o cenário atual, sem valores de referência genéricos, mais esforços devem ser aplicados para encontrar esses valores. Se existirem valores genéricos, os conceitos de Engenharia podem ser aplicados de forma adequada para diagnosticar problemas de *software*, comparando as medidas com os valores de referência. Se os valores genéricos ou universais não existem, talvez possam existir valores genéricos por domínio de aplicação ou tecnologia utilizada? Se sim, a Engenharia poderá ser utilizada de forma adequada.

Se não for possível determinar os valores de referência ou limites, é preciso considerar perguntas, como: encontrar valores de referência é um achado impossível ou as técnicas e modelagem estão sendo utilizadas de maneira inapropriadas?

Muitos dos valores de referência encontrados nos artigos estudados devem ser utilizados com cautela, pois, por exemplo, ou eles foram validados, ou não podem ser replicados, ou podem ser específicos para um determinado tipo de sistema, e não genérico. Também, é importante considerar como uma medida é interpretada ou implementada e qual o impacto que isso causa em valores de referência. Por exemplo, quando a medida LOC é aplicada, é necessário definir como as linhas em branco, comentários e declarações em mais de uma linha, será contado.

É provável que valores de referência, para algumas medidas, sejam diferentes em linguagens de programação diferentes. Isto é, dependente da tecnologia. Por exemplo, linguagens OO podem precisar, intrinsecamente, de diferentes números de classes, ou profundidades de herança diferentes em virtude de características, tais como: *innerclass* e herança múltipla. Seriam necessários valores de referência diferentes. Outras diferenças de recursos podem levar a diferentes números de atributos, métodos, e assim por diante.

Assim, é recomendado o desenvolvimento de um protocolo de pesquisa, a fim de promover a investigação consistente e troca de *'big data'*, entre muitos

pesquisadores. Isso seria semelhante ao que aconteceu em outras áreas, como: biologia (por exemplo, genoma), física de partículas, dentre outras.

Embora a Engenharia de *Software* tenha a necessidade de valores de referência, é importante ressaltar que algumas medidas não possuem um limite natural, um sistema complexo teria sempre valores mais elevados. Por exemplo, isso se aplicaria às medidas LOC, Número de Classes, Número de Métodos e Número de Atributos em um sistema. Para tais medidas, o valor de referência se aplica a médias ou medianas por unidade, em que a unidade pode ser a classe.

O cenário atual conduz a comunidade de Engenharia de *Software* a investir mais esforços para estabelecer os valores de referência, a fim de transformar a Engenharia de *Software* em Engenharia mensurável e com qualidade que pode ser compreendida e explicada por números confiáveis.

5 PRÁTICA DE MERCADO DE *SOFTWARE* LIVRE

Neste Capítulo, são apresentados os resultados relacionados às práticas de mercado de *software* livre. Inicialmente, apresentam-se as medidas, os sistemas de *software* a serem analisados e a ferramenta de coleta selecionada. Em seguida, apresentam-se a coleta dos dados e os resultados das análises dos dados obtidos. Por fim, são apresentados seis fatores que impactam na medição e na análise de medidas.

5.1 Selecionar Medidas

Após a condução da RSL, foi constatado que as medidas propostas por CK (CHIDAMBER; KEMERER, 1994) são as medidas mais estudadas e referenciadas na literatura. Em 10 dos 19 artigos encontrados na RSL, apresentam-se estudos relacionados às medidas de CK. Além de serem as medidas mais populares da literatura, abrangem importantes fatores de qualidade, como: acoplamento, coesão, complexidade, tamanho e herança.

Portanto, foram selecionadas para serem analisadas as seis medidas de CK: WMC, DIT, NOC, CBO, RFC e LCOM2. Além disso, foram selecionadas as medidas Ca e Ce usadas por CBO e duas medidas de tamanho: LOC e nCL para verificar como as medidas de CK se comportam quando comparadas a ela. No total, 10 medidas foram selecionadas e estudadas, para as quais foi possível fazer coleta automatizada, pois havia ferramentas disponíveis.

5.2 Selecionar Ferramenta de Coleta

Para realizar a coleta das medidas, foi selecionada a ferramenta CKJM Ext (JURECZKO; SPINELLIS, 2010) porque implementa, adequadamente, as

medidas da suíte CK. Além disso, a ferramenta possui documentação que descreve as limitações e as observações referentes às medidas implementadas.

CKJM *Ext* (JURECZKO; SPINELLIS, 2010) é uma extensão da ferramenta CKJM (SPINELLIS, 2005). Conforme Jureczko e Spinellis (2010), essa é a versão estendida do CKJM 1.8 e a principal diferença entre a versão original é a inclusão de novas medidas de *software* calculadas.

CKJM Ext é muitiplataforma e não possui interface gráfica. As instruções são passadas por linhas de comando. A ferramenta disponibiliza documentação com as instruções de como executar, como personalizar o *output*, descrição das medidas e detalhes da implementação.

Em CKJM Ext, são permitidos arquivos no formato .jar ou .class como entrada e, como saída produz arquivos no formato de texto. Nessa versão, houve uma simplificação no processo de transformar os arquivos de saída em .XML e, assim como na ferramenta CKJM, cada linha do arquivo de saída representa uma classe e, em cada linha, contém o nome completo da classe e os valores calculados da classe correspondente. A extração das informações para processamento e cálculo das medidas é feita utilizando a biblioteca *Apache Commons BCEL*.

Atualmente, essa ferramenta encontra-se na versão 2.1 de 2011, calcula as medidas da suíte CK, algumas medidas da suíte QMOOD e outras medidas de qualidade. As medidas são: *Number of Public Methods for a class (NPM); Lack of cohesion in methods (LCOM 3 Henderson-Sellers version); Lines of Code (LOC); Data Access Metric (DAM); Measure of Aggregation (MOA); Measure of Functional Abstraction (MFA); Cohesion Among Methods of Class (CAM); Inheritance Coupling (IC); Coupling Between Methods (CBM); Average Method Complexity (AMC); McCabe's Cyclomatic Complexity (CC).*

Uma descrição, conforme a ferramenta CKJM Ext, das medidas utilizadas neste trabalho, é apresentada no Quadro 5 (Apêndice C).

5.3 Selecionar Sistemas de *Software* a Serem Analisados

Os dados utilizados neste estudo são 3 versões, de 107 sistemas de *software* de *opensource*, desenvolvidos em Java, obtidos pelo *SourceForge* (SOURCEFORGE, 2012), em suas últimas versões acima do ano de 2010. No total, 237.370 classes foram coletadas com os sistemas de *software* selecionados, sendo: 53.770 classes de *software* denominadas versão 1; 77.513 classes para os sistemas de *software* denominados versão 2; e 106.087 classes dos sistemas de *software* denominados versão 3.

A base *SourceForge* (SOURCEFORGE, 2012) foi utilizada por ser um dos maiores repositórios de sistemas de *software opensource* da web. Atualmente, são disponibilizados, aproximadamente, 430.000 projetos, sendo a linguagem Java a mais frequente em projetos, com, aproximadamente, 52.151 disponíveis. O *SourceForge* vem sendo utilizado por diversos estudos relacionados à medição de *software* (DALLAL, 2012; FERREIRA et al., 2009, 2011; LINCKE; LUNDBERG; LÖWE, 2008).

A seleção dos sistemas de *software* no *SourceForge* foi aleatória. Entretanto, os sistemas de *software* selecionados seguiam os seguintes critérios:

i) desenvolvido em Java;
ii) disponibilidade do *bytecode*;
iii) possuir três versões disponíveis; e
iv) versão mais recente acima de 2010.

Os Critérios I e II foram definidos em virtude do fato de a ferramenta de coleta selecionada utilizar o *bytecode* de sistemas de *software* desenvolvidos na linguagem Java para efetuar as medições. O Critério III foi definido com o intuito de verificar se existem diferenças significativas entre as três versões de um determinado *software* e analisar o comportamento das medidas considerando as versões avaliadas. Por meio dessa análise, foi possível verificar se houve crescimento ou decrescimento entre os valores das medidas nas versões

selecionadas. O Critério IV foi estabelecido com o intuito de estabelecer valores de referência atuais, baseados na prática de mercado de *software* livre.

Foram selecionadas três versões para verificar o comportamento médio das medidas selecionadas. Objetivou-se, com isso, verificar se uma determinada medida apresenta algum comportamento padrão médio.

Para a seleção das versões, buscou-se baixar as últimas três versões de cada *software*, evitando as versões intermediárias. Entretanto, muitos sistemas de *software* encontrados estavam abaixo da versão 3.0. Dessa forma, foi feito o *download* da versão equidistante entre a primeira versão e a versão mais recente. A escolha das versões foi dificultada por não haver um padrão de versionamento no *SourceForge*. Foram priorizadas as versões não intermediárias por trazerem alterações de maior impacto quando comparado aos sistemas de *software* de versões intermediárias.

Os projetos de *software* foram classificados em 25 domínios de aplicação, sugeridos pelo próprio *SourceForge*. Vale ressaltar que projetos mantidos e disponibilizados pelo *SourceForge* são classificados pelos próprios mantenedores dos sistemas. Logo, é difícil afirmar se um *software* pertence ao domínio classificado. Entretanto, trabalhos como: utilizam a mesma classificação para determinar valores de referência (FERREIRA et al., 2011).

Grande parte dos sistemas de *software* identificados foram sistemas de alta relevância por ter uma taxa de *download* alta. Foi verificado, também, que grande parte dos sistemas de *software*, com baixa classificação, não apresentavam 3 versões ou, na maioria das vezes, a versão mais recente era anterior a 2010.

Na Tabela 58 (Apêndice D) apresentam-se os sistemas de *software*, as versões, as datas de lançamento de cada versão, a quantidade de classes presente em cada versão e os domínios de aplicação.

5.4 Coletar Dados

Após fazer o levantamento das 3 versões dos 107 sistemas de *software*, deu-se início ao processo de medição. No total, foram emitidos pela ferramenta CKJM Ext 321 relatórios. Cada relatório foi conferido por um membro do Grupo de Pesquisa em Engenharia de *Software* (PqES-UFLA)[1]. Foram verificadas duplicidades e anormalidades nos valores, objetivando minimizar os erros e garantir a qualidade do trabalho. Quando havia dúvida por parte do revisor, o *software* era novamente medido e comparado com o anterior.

Feita a conferência, os dados foram organizados conforme o planejamento das análises. Visando eliminar eventuais erros humanos, o processo de organização dos dados foi automatizado. Ou seja, os 321 relatórios foram estruturados na tabela final de forma automática.

Por fim, a tabela final foi conferida por um revisor que selecionou, aleatoriamente, aproximadamente, 30% dos sistemas *software* da tabela final. Esses 30% foram conferidos cautelosamente. Como não houve erros na conferência dessa amostra, deu-se inicio ao processo de análise.

As medidas coletadas foram organizadas de forma a prover facilidades no momento das análises estatísticas. A planilha foi estruturada em colunas, em que: a coluna um contém um índice para o domínio de aplicação. A coluna dois contém um índice para o *software*, na coluna três, apresentam-se as versões e, por fim, nas outras colunas foram apresentadas as medidas coletadas.

5.5 Análises Estatísticas

Nesta seção, são apresentados os resultados das análises estatísticas realizadas nos dados coletados. Inicialmente, são apresentadas as estatísticas descritivas dos dados. Em seguida, apresentam-se os seguintes resultados: a)

[1] PqES/ UFLA - http://pesquisa.dcc.ufla.br/pqes/

análise da dinâmica entre as versões por simples comparação; b) testes de média, aplicando teste não paramétrico de *Kruskall-Wallis;* c) intervalos das medidas praticadas no mercado de *software* livre*;* e d) análise de correlação entre as medidas selecionadas.

5.5.1 Estatística Descritiva dos Dados

Nas Tabelas 36, 37 e 38 apresentam-se, para cada medida selecionada, as estatísticas descritivas geradas com base nos dados coletados. Na Tabela 36, apresentam-se as estatísticas descritivas dos sistemas de *software* na versão 1 (53.770 classes); na Tabela 37, apresentam-se as estatísticas descritivas dos sistemas de *software* na versão 2 (77.513 classes); na Tabela 38, apresentam-se as estatísticas descritivas dos sistemas de *software* na versão 3 (106.087 classes). No total, 237.370 classes referentes às 3 versões de 107 sistemas de *software opensource*, desenvolvidos na linguagem JAVA, foram analisados.

Nas Tabelas 36, 37 e 38 é possível observar que a distribuição dos dados é assimétrica, pois a média e a mediana das medidas avaliadas são diferentes. Além disso, o coeficiente de curtose maior do que zero indica que a distribuição tem uma cauda direita mais pesada e possui valores acima da média. Também, é possível observar variância elevada, indicando que grande parte dos valores não são semelhantes, principalmente, nas medidas LOC, LCOM2 e nCL, em que a discrepância entre os valores coletados foi maior, nas três versões avaliadas.

Verifica-se, considerando as três versões analisadas, que as medidas DIT, NOC e Ce foram as que apresentaram o menor desvio padrão, indicando que os dados tiveram dispersão baixa, quando comparado com o desvio padrão das demais medidas. A medida LCOM2 foi a que apresentou maior dispersão dos dados, fato que pode ser comprovado analisando os desvios padrão presentes nas Tabelas 36, 37 e 38.

Tabela 36 Estatística descritiva das classes da versão 1 de 107 sistemas de *software* [N= 53.770].

		WMC	DIT	NOC	CBO	RFC	LCOM2	Ca	Ce	LOC	nCL
Media		8	0.9	0.3	8.6	22.7	143.4	4.5	4.9	217.7	502.5
Mediana		4	1	0	4	10	1	1	3	49	193
Moda		2	1	0	1	4	0	1	1	14	10
Desvio Padrão		17.9	1.2	5.6	20.7	42.8	4216.6	18.8	8.4	902.7	942
Variância		321.4	1.5	31.1	428.4	1833.1	17779837.2	352.6	71.3	814926.5	887502.4
Assimetria		20	2.3	144.7	23.2	12.9	94.5	29	13.1	61.5	4.8
Curtose		761.8	6.3	25138.8	927.6	416.6	10952.6	1301.1	466.9	7519.7	0.2
Min		0	0	0	0	0	0	0	0	0	2
Max		1094	8	1046	1146	2614	588409	1139	509	126513	7594
Soma		429897	55747	16207	464371	1222425	7709241	241642	265795	11707123	53767
Percentis	**10**	2	0	0	1	3	0	0	0	9	27
	20	2	0	0	2	4	0	1	1	14	70
	25	2	0	0	2	5	0	1	1	18	81
	30	2	0	0	3	6	0	1	1	22	96
	40	3	1	0	3	8	0	1	2	32	140
	50	4	1	0	4	10	1	1	3	49	193
	60	5	1	0	6	14	3	1	4	79	298
	70	7	1	0	8	21	8	2	5	128	368
	75	8	1	0	9	25	13	3	6	165	457
	80	10	1	0	11	31	21	4	7	221	532
	90	17	2	0	18	52	76	9	11	467	1252

(*) nCL é uma medida calculada em nível de *software*, logo, a estatística descritiva foi calculada para N=107.

Tabela 37 Estatística descritiva das classes da versão 2 de 107 sistemas de *software* [N= 77.513].

		WMC	DIT	NOC	CBO	RFC	LCOM2	Ca	Ce	LOC	nCL
Media		7.9	0.9	0.3	9.1	22.7	132.5	4.7	5.2	215.1	724.4
Mediana		4	1	0	5	10	1	1	3	49	339
Moda		2	1	0	1	4	0	1	1	14	98
Desvio Padrão		17.2	1.1	5	21	42.2	3715.5	18.9	8.5	851	1179.7
Variância		296	1.3	25.4	441.6	1783.9	13805660.2	360.1	73.2	724343.8	1391748.7
Assimetria		19.2	2.4	140.9	19.1	11.9	99.9	24.1	9.7	53.6	3.9
Curtose		747.5	7.5	26270.1	679.8	355.6	12384.3	971.9	257.8	6634.6	19,9
Min		0	0	0	0	0	0	0	0	0	6
Max		1080	8	1046	1146	2548	572148	1139	468	126513	8353
Soma		615123	76374	23799	711832	1762109	10274344	366555	407307	16675128	77514
Percentis	**10**	2	0	0	1	3	0	0	0	8	56
	20	2	0	0	2	4	0	1	1	14	112
	25	2	0	0	2	5	0	1	1	18	126
	30	2	0	0	3	6	0	1	1	22	163
	40	3	1	0	4	8	0	1	2	32	225
	50	4	1	0	5	10	1	1	3	49	339
	60	5	1	0	6	14	3	2	4	76	455
	70	7	1	0	8	20	8	2	5	123	660
	75	8	1	0	10	25	13	3	6	161	771
	80	10	1	0	11	31	21	4	7	215	957
	90	17	2	0	18	53	76	9	12	461	1823

(*) nCL é uma medida calculada em nível de *software*, logo, a estatística descritiva foi calculada para N=107.

Tabela 38 Estatística descritiva das classes da versão 3 de 107 sistemas de *software* [N= 106.087].

		WMC	DIT	NOC	CBO	RFC	LCOM2	Ca	Ce	LOC	nCL
Media		8	0.9	0.3	9.3	22.6	149.2	4.8	5.3	215.2	991.1
Mediana		4	1	0	5	10	1	1	3	49	364
Moda		2	1	0	2	4	0	1	1	14	15
Desvio Padrão		19.1	1.1	5	22.1	41.7	7016.7	20	8.5	838.6	1643.7
Variância		368	1.2	25	489.4	1741.7	49234611.3	403.5	72.9	703272.5	2702060.8
Assimetria		38.4	2.5	131.9	18.4	13.2	216.5	22.7	9.1	45.8	3.8
Curtose		3759.9	8.1	23918.5	600.9	483.4	56630.9	833.3	228.9	5317	18.6
Min		0	0	0	0	0	0	0	0	0	9
Max		2617	8	1066	1172	2637	1943164	1164	482	126513	11500
Soma		850482	101259	33324	990819	2406507	15830003	510303	562898	22839754	106056
Percentis	**10**	2	0	0	1	3	0	0	0	8	97
	20	2	0	0	2	4	0	1	1	15	156
	25	2	0	0	2	5	0	1	1	19	184
	30	2	0	0	3	6	0	1	1	22	196
	40	3	1	0	4	8	0	1	2	33	266
	50	4	1	0	5	10	1	1	3	49	364
	60	5	1	0	6	14	3	2	4	76	574
	70	7	1	0	8	20	7	2	5	123	870
	75	8	1	0	9	25	12	3	6	161	1004
	80	10	1	0	11	31	21	4	7	216	1390
	90	17	2	0	19	53	75	9	12	460	2490

(*) nCL é uma medida calculada em nível de *software*, logo, a estatística descritiva foi calculada para N=107.

Nas Tabelas 36, 37 e 38, também, são apresentados os valores mínimo (Min) e máximo (Max) que representam, respectivamente, o menor e maior valor dentre todas as medidas analisadas. Por exemplo, considerando a medida WMC na versão 3, é possível observar que Min=0 e Max=2617. Ou seja, houve classes em que o número de métodos foi igual a 0 e classes com 2617 métodos.

Por fim, nas Tabelas 36, 37 e 38, são apresentados os valores dos percentis para cada medida. Os percentis dividem a amostra ordenada em partes, cada parte com porcentagem de dados, aproximadamente, iguais. Por exemplo, verifica-se, na versão 3 da medida WMC, que 30% dos valores são menores ou iguais a 2. Também é possível observar que 75% dos dados são menores ou iguais a 8.

Na Tabela 39 são apresentados, para cada versão, os intervalos sem *outliers*, com *outliers* suaves e *outliers* extremos das medidas selecionadas. Também é apresentada a quantidade de classes com valores entre os intervalos identificados.

Os limites superiores (LSout) e inferiores (LIout) dos intervalos de *outliers*, foram definidos da seguinte forma:

LIout = Q3 + 1.5(Q3 – Q1)

LSout = Q3 + 3(Q3 – Q1)

Os valores que estiverem entre os LIout e LSout são considerados *outliers* suaves. Os valores abaixo de LIout, definem um intervalo sem *outlier*s e os valores acima de LSout definem os *outliers* extremos (MOROCO, 2007).

Vale ressaltar que expressões utilizadas para delimitar os *outliers* são, de certo modo, arbitrárias e podem não delimitar adequadamente. Conforme Triola (2005), caso existam *outliers*, deve-se verificar se são provenientes de erros de digitação e mensuração. Ainda, é importante salientar que nem todo *outlier* é um erro. Alguns são valores verdadeiros.

Na Tabela 39, é possível observar que os intervalos sem *outliers* da medida WMC variam de 0 a 16 e os intervalos com *outliers* suaves variam de 17 a 26 nas três versões estudadas. Os *outliers* extremos variam de 27 a 1.094 na versão 1, de 27 a 1.080 na versão 2 e de 27 a 2.617 na versão 3. No total, aproximadamente, 90% das classes encontram-se no intervalo sem *outliers*, 5% no intervalo com *outliers* suaves e 5% no intervalo de *outliers* extremos nas três versões estudadas. As quantidades de classes entre os intervalos apresentados podem ser visualizados na Tabela 39. Por fim, também, é possível observar que a medida WMC possui assimetria positiva (à direita), nas três versões avaliadas, dado que a média aritmética é superior à mediana.

Tabela 39 *Outliers* das medidas considerando as três versões estudadas.

Medidas	**Vers.**	**Intervalo sem *outliers***	**Intervalo com *outliers* suaves**	**Intervalo com *outliers* extremos**	**Qtd de classes no intervalo sem *outliers***	**Qtd de *outliers* suaves**	**Qtd de *outliers* extremos**
WMC	1	[0, 16]	[17, 26]	[27, 1094]	48263	2796	2711
	2	[0, 16]	[17, 26]	[27, 1080]	69525	3946	4042
	3	[0, 16]	[17, 26]	[27, 2617]	94834	5510	5743
DIT	1	[0, 2]	[3, 4]	[5, 8]	49754	1503	2513
	2	[0, 2]	[3, 4]	[5, 8]	72617	1809	3087
	3	[0, 2]	[3, 4]	[5, 8]	99900	2474	3713
NOC	1		0		-	-	-
	2		0		-	-	-
	3		0		-	-	-
RFC	1	[0, 54]	[55, 85]	[86, 2614]	48720	2640	2410
	2	[0, 54]	[55, 85]	[86, 2548]	70178	3778	3557
	3	[0, 54]	[55, 85]	[86, 2637]	95763	5451	4873
CBO	1	[0, 19]	[20, 30]	[31, 1146]	49198	2358	2214
	2	[0, 21]	[22, 34]	[35, 1146]	71416	3213	2884
	3	[0, 19]	[20, 30]	[31, 1172]	96187	5059	4841
LCOM2	1	[0, 32]	[33, 52]	[53, 588409]	45006	1849	6915
	2	[0, 32]	[33, 52]	[53, 572148]	65084	2538	9891
	3	[0, 29]	[30, 48]	[49, 1943164]	88445	4205	13437
Ca	1	[0, 5]	[6, 9]	[10, 1139]	45476	3349	4945
	2	[0, 5]	[6, 9]	[10, 1139]	65250	4928	7335
	3	[0, 5]	[6, 9]	[10, 1164]	89524	6704	9859
Ce	1	[0, 13]	[14, 21]	[22, 509]	49927	2305	1538
	2	[0, 13]	[14, 21]	[22, 468]	71371	3647	2495
	3	[0, 13]	[14, 21]	[22, 482]	97459	5012	3616

Pode-se notar, analisando a Tabela 39, que os intervalos sem *outliers* da medida DIT variam de 0 a 2, o intervalo com *outliers* suaves variam de 3 a 4 e o intervalo de *outliers* extremos variam de 5 a 8 nas três versões analisadas. No total, aproximadamente, 93% das classes encontram-se no intervalo sem *outliers*, 3% no intervalo com *outliers* suaves e 4% no intervalo de *outliers* extremos nas três versões estudadas. As quantidades de classes entre os intervalos apresentados podem ser visualizados na Tabela 39. Por fim, conclui-se que a medida DIT possui assimetria negativa (à esquerda), nas três versões estudadas, dado que a média aritmética é levemente inferior à mediana.

Observa-se, na Tabela 39, que os intervalos sem *outliers*, com *outliers* suaves e *outliers* extremos, na medida NOC, foi 0 nas três versões avaliadas. Tal comportamento é por causa da grande quantidade de zeros na amostra, indicando que as maiorias das classes não possuem classes subordinas. Por fim, conclui-se que a medida NOC possui uma leve assimetria positiva (à direita), nas três versões avaliadas, dado que a média aritmética é superior à mediana.

Na Tabela 39 é possível observar que os intervalos sem *outliers* da medida RFC variam de 0 a 54 e os intervalos com *outliers* suaves variam de 55 a 85 nas três versões analisadas. Os *outliers* extremos variam de 86 a 2614 na versão 1, de 86 a 2548 na versão 2 e de 86 a 2637 na versão 3. No total, aproximadamente, 90% das classes encontram-se no intervalo sem *outliers*, 5% no intervalo com *outliers* suaves e 5% no intervalo de *outliers* extremos nas três versões estudadas. As quantidades de classes que estão entre os intervalos apresentados podem ser visualizados na Tabela 39. Por fim, conclui-se que a medida RFC possui assimetria positiva (à direita), nas três versões analisadas, dado que a média aritmética é superior à mediana.

Pode-se notar que os intervalos sem *outliers* da medida CBO variam de 0 a 19 nas versões 1 e 3, e de 0 a 21 na versão 2. Os intervalos com *outliers* suaves variam de 20 a 30 nas versões 1 e 3, e na versão 2 varia de 22 a 34. Os

outliers extremos variam de 31 a 1146 na versão 1, de 35 a 1146 na versão 2 e de 31 a 1172 na versão 3. No total, aproximadamente, 91% das classes encontram-se no intervalo sem *outliers*, 4% no intervalo com *outliers* suaves e 4% no intervalo de *outliers* extremos nas três versões estudadas. As quantidades de classes que estão entre os intervalos apresentados podem ser visualizadas na Tabela 39. Por fim, conclui-se que a medida CBO possui assimetria positiva (à direita), nas três versões avaliadas, dado que a média aritmética é superior à mediana.

Observa-se, na Tabela 39, que os intervalos sem *outliers* da medida LCOM2 variam de 0 a 32 nas versões 1 e 2, e de 0 a 29 na versão 3. Os intervalos com *outliers* suaves variam de 33 a 52 nas versões 1 e 2, e de 30 a 48 na versão 3. Os *outliers* extremos variam de 53 a 588409 na versão 1, de 53 a 572148 na versão 2 e de 49 a 1943164 na versão 3. No total, aproximadamente, 84% das classes encontram-se no intervalo sem *outliers*, 4% no intervalo com *outliers* suaves e 12% no intervalo de *outliers* extremos nas três versões estudadas. Por fim, conclui-se que a medida LCOM2 possui uma forte assimetria positiva, nas três versões analisadas, dado que a média aritmética é superior à mediana.

Na Tabela 39 é possível observar que os intervalos sem *outliers* da medida Ca variam de 0 a 5 e os intervalos com *outliers* suaves variam de 6 a 9 nas três versões estudadas. Os *outliers* extremos variam de 10 a 1139 nas versões 1 e 2 e de 10 a 1164 na versão 3. No total, aproximadamente, 84% das classes encontram-se no intervalo sem *outliers*, 6% no intervalo com *outliers* suaves e 9% no intervalo de *outliers* extremos nas três versões estudadas. As quantidades de classes que estão entre os intervalos apresentados podem ser visualizadas na Tabela 39. Por fim, conclui-se que a medida Ca possui assimetria positiva (à direita), nas três versões analisadas, dado que a média aritmética é superior à mediana.

Na Tabela 39 é possível observar que os intervalos sem *outliers* da medida Ce variam de 0 a 13 e os intervalos com *outliers* suaves variam de 14 a 21 três versões estudadas. Os *outliers* extremos variam de 22 a 509 na versão 1, de 22 a 468 na versão 2 e de 22 a 482 na versão 3. No total, aproximadamente, 92% das classes encontram-se no intervalo sem *outliers*, 5% no intervalo com *outliers* suaves e 3% no intervalo de *outliers* extremos nas três versões estudadas. As quantidades de classes que estão entre os intervalos apresentados podem ser visualizadas na Tabela 39. Por fim, conclui-se que a medida Ce possui assimetria positiva (à direita), nas três versões analisadas, dado que a média aritmética é superior à mediana.

As medidas LOC e nCL não possuem limites naturais, consequentemente, quanto maior o sistema maior a quantidade de linhas e/ou maior a quantidade de classe. Seria arbitrário afirmar que tais medidas possuem *outliers* visto que são dependentes do tamanho do sistema.

5.5.2 Dinâmica Entre as Versões

Na Tabela 40, é apresentada uma dinâmica entre as versões estudadas, considerando a média das medidas coletadas. São apresentadas 7 situações (Figura 4): A primeira (#1) representa a porcentagem de sistemas de *software* que apresentaram crescimento médio contínuo das medidas avaliadas entre as 3 versões, ou seja, V1 < V2 < V3; a segunda (#2) representa a porcentagem dos sistemas de *software* que tiveram decrescimento médio contínuo das medidas avaliadas entre as 3 versões, ou seja, V1 > V2 > V3; a terceira (#3) situação indica a porcentagem dos sistemas de *software* que apresentaram crescimento de V1 para V2, mas de V2 para V3, decresceu; na quarta (#4) situação, é apresentada a porcentagem dos sistemas de *software* que decresceram da V1 para a V2, mas da versão V2 para V3, cresceu; nas situações cinco (#5), seis

(#6) e sete (#7), respectivamente, é apresentada a porcentagem dos sistemas de *software* igual entre as versões V1 e V2, mas V2 ≠ V3, os sistemas de *software* diferentes entre V1 ≠ V2, mas iguais entre V2 e V3 e, por fim, os sistemas de *software* que apresentaram a média das medidas igual nas três versões avaliadas.

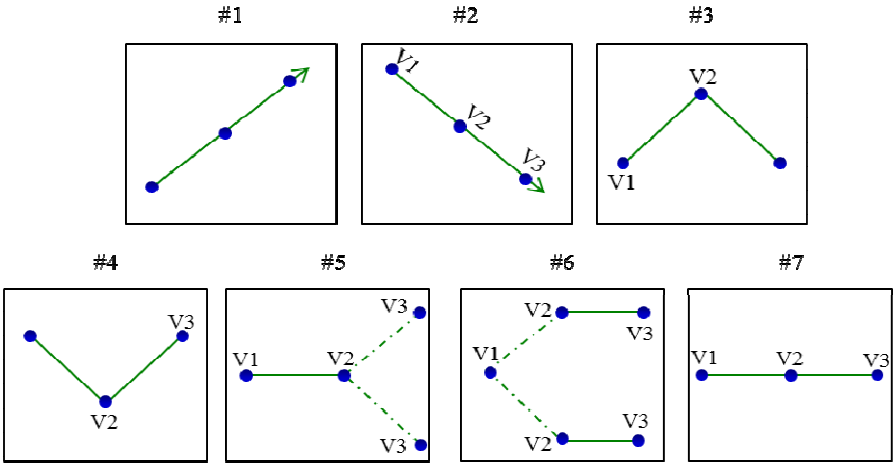

Figura 4 Situações identificadas na dinâmica entre as versões.

Na situação #1, verifica-se que a medida nCL, em 72% dos sistemas de *software*, aumentou continuamente considerando as três versões analisadas. Esse aumento pode ser em virtude da inserção de novas funções, conforme os sistemas de *software* foram sendo atualizados. Também é possível observar que as medidas CBO, LCOM2, Ca e Ce aumentaram, continuamente, em mais de 40% dos sistemas de *software*, nas três versões analisadas. A medida DIT foi a que obteve menor porcentagem de *software* com crescimento médio contínuo, com 13.1% dos sistemas de *software* avaliados.

Tabela 40 Dinâmica entre as versões de um mesmo *software*.

#	Situações	nCl	WMC	DIT	NOC	CBO	RFC	LCOM2	Ca	Ce	LOC
1	V1 < V2 < V3	72.0%	29.0%	13.1%	30.0%	44.9%	24.3%	43.9%	43.9%	42.1%	31.8%
2	V1 > V2 > V3	1.9%	23.4%	34.0%	23.0%	13.1%	20.6%	17.8%	13.1%	13.1%	19.6%
3	V1 < V2 > V3	14.0%	26.2%	26.2%	25.0%	26.2%	29.9%	23.4%	26.2%	28.0%	24.3%
4	V1 > V2 < V3	6.5%	21.5%	24.2%	17.0%	14.0%	25.2%	15.0%	15.0%	15.0%	24.3%
5	V1 = V2 ≠ V3	3.7%	0.0%	0.6%	1.0%	0.0%	0.0%	0.0%	0.9%	0.0%	0.0%
6	V1 ≠ V2 = V3	1.9%	0.0%	1.0%	0.0%	2.0%	0.0%	0.0%	0.9%	1.9%	0.0%
7	V1 = V2 = V3	0.0%	0.0%	1.0%	4.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	∑	100%	100%	100%	100.1%	100%	100.1%	100%	100%	100%	100%

Na situação #2, é possível observar, analisando a Tabela 40, que em 1.9% dos sistemas de *software*, a medida nCL apresentou um decrescimento médio contínuo, considerando as três versões avaliadas. Verifica-se, também, que em 34% dos sistemas *software* avaliados, considerando a medida DIT, houve decrescimento médio contínuo. Ou seja, em 34% dos sistemas de *software* houve uma diminuição do uso de Herança, conforme foram sendo atualizados.

Nas situações #3 e #4, é possível observar alterações nas versões intermediárias de cada *software*. Na situação #3, verifica-se que em mais de 20% dos sistemas de *software* avaliados, considerando a média das medidas, exceto a medida nCl, V2 apresentou valor médio superior quando comparada com as versões V1 e V3. Enquanto, na situação #4, V2 apresentou valores médios inferiores a V1 e V3.

Nas situações (#5, #6 e #7) foi aplicado o teste não paramétrico de *Kruskall-Wallis*, com significância de 5%, a fim de verificar se as médias das versões são estatisticamente iguais ou diferentes.

Na situação #5, é possível observar a porcentagem dos sistemas de *software* que não apresentaram diferenças entre as médias das versões V1 e V2, mas apresentam diferença entre a média das versões V2 e V3. Observa-se que em 3.7% dos sistemas de *software*, na medida nCL, são estatisticamente iguais, pois não sofreram alterações entre as versões V1 e V2. Em 97.3% dos sistemas

de *software* analisados, houve alteração estrutural (inserção ou remoção de classes) entre V1 e V2. De forma similar, a situação #6 indica a porcentagem dos sistemas de *software* que não apresentaram variações médias, entre as medidas, considerando as V2 e V3. Verifica-se que as medidas nCL, CBO e Ce, em, aproximadamente, 98% dos sistemas de *software*, sofreram alterações, portanto 2% dos *software* não sofreram. Por fim, na situação #7, apresentada na Tabela 40, é possível verificar a porcentagem dos sistemas de *software* que não apresentaram diferença entre as 3 versões. Observa-se que apenas 1% dos sistemas de *software* da medida DIT e 4% da medida NOC não variaram conforme a versão.

De um modo geral, apenas a medida nCL apresentou um crescimento médio continuo de 72%. Ou seja, o número de classes aumentou em 72% dos sistemas de *software* estudados conforme foram sendo atualizados. Nas demais medidas, um padrão médio não foi identificado.

5.5.3 Testes de Média

Para identificar o teste adequado, realizou-se uma análise preliminar, com o intuito de inferir se as distribuições dos dados seguem uma distribuição normal. Para isso, foi aplicado o teste de normalidade de *Kolmogorov-Sminorv* nos dados de cada uma das medidas selecionadas. Nesse teste, foi constatado que os dados não são modelados por uma distribuição normal, violando os pressupostos da ANOVA (Análise de Variância), popularmente usada em comparações de médias.

Posteriormente, aplicou-se uma transformação logarítmica nos dados para estabilizar a variância e realizou-se uma ANOVA. Entretanto, após realizar a ANOVA, verificou-se, por meio de uma análise de resíduos, a não normalidade dos resíduos. Dessa forma, para os testes de média, foi utilizado o teste não paramétrico de *Kruskall-Wallis.*

O teste de *Kruskall-Wallis*, conhecido como teste H, é um teste não paramétrico que pode ser usado para comparar três ou mais amostras independentes não pareadas, quando os pressupostos de uma Análise de Variância (ANOVA) são violados (KRUSKALL; WALLIS, 1952). Com o teste de *Kruskal-Wallis*, não é necessário fazer suposições a respeito das distribuições que modelam os dados, ele não impõe restrições sobre as comparações realizadas (HEIMAN, 2011).

Para verificar a existência de diferença significativa, para cada medida, entre as três versões de cada *software*, a seguinte hipótese foi elaborada:

H_0: Não existe diferença significativa entre as medidas selecionadas, considerando versões diferentes do mesmo *software*.

H_1: Pelo menos um *software* possui diferença significativa.

Na Tabela 41, é apresentado um resumo da aplicação do teste de *Kruskall-Wallis*, com significância estatística de 5%, nas amostras coletadas. São apresentadas a quantidade, a porcentagem e os sistemas *software* que tiveram diferenças significativas entre as médias das suas versões (*p-value* < 0.05). Os rótulos (S1, S2, S3, ..., S107) apresentados na coluna *Software* da Tabela 41 indica os sistemas de *software* listados e mapeados na Tabela 58 (Apêndice D).

Pelo teste não paramétrico de *Kruskall-Wallis* demonstraram-se diferenças estatisticamente diferentes nas comparações realizadas entre as médias das 3 versões de um mesmo *software*.

Analisando a Tabela 41, verifica-se que a medida NOC apresentou diferenças significativas em 9% dos sistemas de *software* avaliados, enquanto a medida DIT apresentou diferença significativa em 42% dos sistemas de *software* avaliados. No total, foram aplicadas 107 comparações para cada medida selecionada. Dessa forma, conclui-se, com base no teste de *Kruskall-Wallis* e

significância de 5%, que existem diferenças significativas entre as versões dos sistemas de *software* avaliados, rejeitando a hipótese nula H_0 e aceitando a hipótese alternativa H_1.

Tabela 41 *Kruskall-Wallis* aplicado a versões de um mesmo *software*.

Medidas	***Kruskall-Wallis***		***Software***
	Qtd.	**%**	
WMC	24	22%	S1, S5, S6, S18, S19, S23, S25, S27, S28, S33, S40, S45, S58, S59, S70, S74, S77, S80, S84, S89, S91, S97, S98 e S102
DIT	45	42%	S1,S2, S3,S4, S5, S6, S8, S16, S18, S19, S21, S23, S25, S27, S28, S30, S32, S33, S36, S40, S41, S45,, S50, S52, S55, S57, S58, S59, S60, S64, S67, S70, S74, S76, S77, S78, S83, S85, S89, S90, S91, S94, S97, S100 e S104
NOC	10	9%	S4, S16, S22, S27, S28, S33, S67, S85, S100 e S104
CBO	33	31%	S1, S3, S4, S6, S8, S11, S16, S18, S23, S25, S27, S32, S36, S40, S45, S49, S52, S53, S58, S67, S69, S74, S80, S82, S85, S89, S90, S91, S92, S97, S102, S103, S104
RFC	23	21%	S3, S4, S6, S8, S18, S19, S23, S25, S27, S33, S40, S45, S58, S60, S67, S70, S74, S85, S89, S91, S98, S102, S104
LCOM2	24	22%	S1, S3, S4, S5, S6, S19, S25, S27, S28, S35, S40, S45, S52, S58, S59, S74, S84, S89, S97, S98, S99, S100, S102, S03
Ca	20	19%	S1, S8 S18, S19, S22, S23, S40, S45, S54, S58, S67, S70, S74, S80, S85, S87, S91, S92, S99, S103
Ce	24	22%	S3, S4, S6, S8, S18, S22, S23, S30, S36, S45, S58, S60, S62, S74, S76, S80, S82, S85, S91, S92, S93, S102, S103, S104
LOC	23	21%	S3, S4, S6, S8, S18, S19, S23, S25, S27, S33, S36, S40, S41, S45, S58, S60, S69, S70, S76, S85, S89, S91, S102

Após realizar o teste de *Kruskall-Wallis*, para comparar a média das três versões de um mesmo *software*, foi aplicado o teste *post-hoc*, isto é, teste de comparação múltipla, em que a média de um *software* é comparada com os demais sistemas de *software*. Na comparação múltipla, também, foi utilizado o teste de *Kruskall-Wallis* com significância estatística de 5%.

Apresenta-se, na Tabela 42, um resumo dos testes de comparações múltiplas. É apresentada, para as medidas selecionadas, a quantidade de *software* que foi estatisticamente igual a todos os outros, o *software* que mais se

diferenciou dos demais e os sistemas de *software* que foram estatisticamente iguais aos demais.

Ao comparar todos os sistemas de *software* entre si, foi possível observar que 2.8% dos sistemas de *software* não apresentam diferença significativa entre si, na versão 1 da medida WMC. Dentre os sistemas de *software* estudados, o que apresentou maior diferença entre os demais foi o *software* S87 que difere de 100 sistemas de *software* (universo de 106 sistemas de *software*). Na versão 2 da medida WMC, verifica-se que 3.8% sistemas de *software* não apresentaram diferença significativa entre si. O *software* que apresentou maior número de diferenças, na versão 2, foi S43. Por fim, na versão 3, é possível observar que 0.9% *software* não apresentaram diferença significativa entre si. Dos sistemas de *software* analisados, o S43 foi o que apresentou maior numero de diferenças.

Dentre as medidas, a NOC foi a que apresentou maior número de sistemas de *software*, estatisticamente, iguais entre si. Na versão 1, 95% sistemas de *software* foram, estatisticamente, iguais. Na versão 2, 84% do foram, estatisticamente, iguais entre si e, na versão 3, 82% dos sistemas de *software*. É possível observar que, para a medida NOC, os sistemas de *software* apresentaram similaridade alta. Entretanto, conforme os sistemas de *software* foram evoluindo, a similaridade entre os sistemas foi diminuindo, passando de 95% para 82%. Essa diminuição pode ser um indicativo de que os recursos de classes filhas podem estar sendo mais utilizado nos sistemas de *software* atuais.

Ao comparar todos os sistemas de *software* entre si, foi verificado que 5.6% dos sistemas de *software* são, estatisticamente, iguais num universo de 106 sistemas de *software*, na versão 1 da medida DIT. O *software* que apresentou maior quantidade de sistemas de *software*, estatisticamente, diferentes foi S93. Na versão 2 da medida DIT, verifica-se que 1.8% sistemas de *software* não apresentaram diferença significativa entre si.

Tabela 42 Quadro resumo do teste *post-hoc* de *Kruskall-Wallis* com significância de 5%.

		WMC	DIT	NOC	CBO	RFC	LCOM2	CA	CE
Versão 1	**Quantidade de soft. estatisticamente iguais**	3	6	101	2	5	7	10	7
	Soft. com maior núm. de diferenças/núm. de diferenças	S87/100	S93/75	S3, S38, S67, S78/3	S3/89	S87/98	S58/88	S58/88	S3/73
	***Sistemas de software* que não tiveram diferenças**	S97, S105, S106	S11, S14, S31, S92, S97, S105	Exceto: S3, S38, S67, S78/3, S41/S48	S11, S105	S11, S73, S97, S105, S106	S2, S11, S14, S37, S97, S105, S106	S11, S14, S24, S31, S37, S64, S73, S97, S105, S106	S11, S12, S24, S31, S97, S105, S106
Versão 2	**Quantidade de soft. estatisticamente iguais**	4	2	90	0	2	3	7	5
	Soft. com maior núm. de diferenças/núm. de diferenças	S43/97	S80/92	S58/10	S45/103	S87/103	S58/99	S34/91	S3/83
	***Sistema de software* que não tiveram diferenças**	S11, S24, S97, S106	S11, S14	Exceto: S26, S36, S53, S65, S74, S78, SS79, S86, S103/1, S3, S25, S27, S38, S41, S60/4, S67/7 e S58/10	NA	S11, S106	S11, S14, S106	S11, S14, S24, S37, S49, S105, S106	S11, S24, S37, S49, S106
Versão 3	**Quantidade de soft. estatisticamente iguais**	1	1	87	0	2	3	4	0
	Soft. com maior núm. de diferenças/núm. de diferenças	S43/103	S80/94	S58/15	S45/105	S87/102	S58/96	S58/100	S45/98
	***Sistemas de software* que não tiveram diferenças**	S106	S14	Exceto: S19, S26, S40, S74, S79, S100/1, S3, S27, S28, S85, S85, S86, S89/2, S67, S78, S103/3, S60/4, S38, S41/5, S25/10 e S58/15	NA	S11, S106	S11, S14, S106	S14, S24, S105, S106	NA

O *software* que apresentou maior número de diferenças, na versão 2, foi S80. Por fim, na versão 3, é possível observar que 0.9% *software* não apresentou diferença significativa entre si. Dos sistemas de *software* analisados, o S80 foi o que apresentou maior numero de diferenças. Da forma similar é a interpretação para as medidas CBO, RFC, LCOM2, Ca e Ce.

Verifica-se que nas medidas WMC, DIT, NOC, Ca e Ce houve uma diminuição dos sistemas de *software*, estatisticamente, iguais nas três versões avaliadas. As medidas CBO, RFC e LCOM2 mantiveram a quantidade de sistemas de *software*, estatisticamente, iguais nas versões 2 e 3, porém, em relação à versão 1, houve uma diminuição. Tal comportamento pode ser em razão da mudança de estilo de programação ou de atualizações nas linguagens de programação.

No Apêndice E, são apresentados os sistemas de software e a quantidade de sistemas de software que foram, estatisticamente, diferentes no teste de comparação múltipla de *Kruskall-Wallis* com significância estatística de 5%.

5.5.4 Intervalos das Práticas de Mercado de *Software* Livre

Nesta Seção, apresentam-se os resultados da aplicação da técnica *Bootstrap* e definem-se e aplicam-se três critérios para os intervalos das práticas de mercado de *software* livre.

5.5.4.1 *Bootstrap*

Bootstrap é uma técnica de reamostragem proposta por Efron em 1979 (EFRON; TIBSHIRANI, 1993), que objetiva substituir análises estatísticas complicadas e duvidosas por métodos computacionais intensivos. A técnica pode ser aplicada tanto em dados não paramétricos quanto em paramétricos para

realizar cálculos estatísticos de interesse. No caso de dados não paramétricos, não é necessário fazer suposições da distribuição que modela os dados (SKYLAR; SMITH, 2003).

Essa técnica é adequada para avaliar, com precisão, um conjunto de dados limitados, que as técnicas convencionais não são precisas ou válidas. Essa técnica vem sendo aplicada em diversas áreas, como: processamento de sinais, simulações, área médica e vem ganhando atenção na área da Engenharia de *Software* (BRIAND; WUST, 2001; PRECHELT; HUNGER, 2001; SKYLAR; SMITH, 2003).

Neste trabalho, a técnica *Bootstrap* foi utilizada para estimar, com precisão, a média dos valores praticados no mercado. Para aplicar a técnica, primeiramente uma amostra mestre foi selecionada. Uma amostra mestre consiste em uma amostra de tamanho *n*, que representa a população da qual foi retirada.

As amostras mestre, neste trabalho, foram os valores das medidas de cada *software*. Ou seja, cada *software* possui uma quantidade de valores referentes às suas classes, para as medidas WMC, DIT, NOC, CBO, RFC, LCOM2, Ca, Ce. Assim, para cada versão dos 107 sistemas de *software*, foi executada a técnica *Bootstrap* utilizando 15.000 repetições com reposição. Além disso, foi utilizado o método de correção de viés acelerado (BCa - *Biased-Corrected Accelerated*), com significâncias de 5%.

Segundo Ferreira (2013), é recomendado 2.000 ou mais repetições para obtenção resultados robustos. O nível de significância de 5% foi adotado por ser um nível de significância usual e comumente utilizado. O método BCa foi utilizado, porque é o método mais indicado, quando a assimetria dos dados estiver presente de maneira muito forte (EFRON; TIBSHIRANI, 1993).

No total, 321 execuções foram realizadas, para cada medida e, em cada execução, foram obtidos: uma média *Bootstrap*, limite inferior e superior, *bias* e

SE. Os limites inferiores e superiores delimitam uma região em que abrange pelo menos 95% das médias das reamostras.

Posteriormente, após obter um conjunto de 107 médias *Bootstrap* para cada medida e versão, uma média global foi calculada. Para cada versão e medida, foi aplicada a técnica *Bootstrap*, usando 15.000 repetições com reposição e significância estatística de 5%. A amostra mestre da média global foi constituída das médias de cada *software*.

Na Tabela 43 apresenta-se um quadro resumo com os resultados obtidos após aplicar a técnica de *Bootstrap*, para cada medida, nas 3 versões dos 107 sistemas de *software*. Na coluna um, são apresentadas nomenclaturas das medidas; na coluna dois, são apresentadas as versões; na coluna três, são apresentadas as médias globais *Bootstrap*; nas colunas quatro e cinco, são apresentados, respectivamente, os limites inferiores e superiores das médias calculadas; na coluna seis, são apresentados os *bias*, que representam a diferença entre as médias das reamostras com a amostra original, um valor pequeno do *bias* é uma indicação de que os valores estimados devem estar próximos dos verdadeiros valores; e, por fim, na coluna sete, é apresentado o SE (*standard error*) erro padrão, isto é, o desvio padrão das medidas reamostradas.

Para aplicar a técnica *Bootstrap*, foi utilizada a ferramenta R (R CORE TEAM, 2014) e o pacote *BootES* disponibilizado por Kirby e Gerlanc (2013), para as reamostragens e análises.

Verifica-se, analisando a Tabela 43, que medida LCOM apresenta o intervalo entre os limites inferiores e superiores mais largos. Uma das causas dessa variação deve-se ao fato da variabilidade dos valores das medidas. Foi observado que, quando há uma discrepância elevada entre os valores das medidas, os SE tendem a ser mais altos, afetando os limites inferiores e superiores.

Tabela 43 *Bootstrap* aplicado à média das 3 versões dos 107 sistemas de *software* (N=107)

Medidas	Versão	Média	LI	LS	Bias	SE
WMC	1	7.688	7.133	8.403	-0.002	0.319
	2	7.784	7.250	8.459	-0.005	0.304
	3	7.930	7.369	8.715	0.003	0.333
DIT	1	1.143	1.062	1.235	-0.001	0.044
	2	1.077	1.000	1.162	0.000	0.041
	3	1.036	0.966	1.116	0.000	0.038
NOC	1	0.241	0.209	0.274	0.000	0.017
	2	0.271	0.239	0.307	0.000	0.017
	3	0.276	0.245	0.310	0.000	0.017
RFC	1	23.240	22.004	24.866	0.002	0.723
	2	23.554	22.208	25.230	0.011	0.767
	3	23.245	21.942	24.795	0.004	0.716
CBO	1	7.144	6.633	7.680	0.000	0.269
	2	7.711	7.179	8.334	0.002	0.294
	3	7.783	7.247	8.367	0.001	0.285
LCOM2	1	85.798	65.861	122.693	-0.133	13.427
	2	100.392	76.349	148.773	-0.310	16.812
	3	148.346	102.470	271.038	-0.239	34.957
Ca	1	3.632	3.373	3.912	-0.001	0.138
	2	3.937	3.666	4.254	0.000	0.149
	3	3.942	3.670	4.242	0.000	0.143
Ce	1	4.278	3.997	4.572	0.000	0.146
	2	4.540	4.243	4.856	0.000	0.157
	3	4.584	4.285	4.899	-0.001	0.156

5.5.4.2 Critérios e Intervalos das Práticas de Mercado de *Software* Livre

Para cada medida estudada, exceto as medidas nCL e LOC, foi gerado um gráfico de frequência absoluta (Apêndice F) para visualizar e entender o comportamento dos valores mais frequentes. Pode-se observar, analisando as Figura 15, 16, 17, 18, 19, 20, 21 e 22 (Apêndice F) que, em todas as medidas, os valores mais frequentes concentram-se próximos ao zero. Além disso, é possível observar que as medidas possuem uma assimetria positiva a direita forte. Dessa forma, as médias dos valores brutos deslocam-se de forma que podem não ser adequado para as análises. Visando suavizar problemas com o deslocamento das

médias, optou-se pelas técnicas *Bootstrap* utilizando o método BCa. Em seguida, foram elaborados três critérios para definir um intervalo ou região de interesse, em cada medida, dos valores praticados no mercado. Os critérios foram:

C1 – Deslocar uma unidade, com base na média global *Bootstrap*, para a direita e esquerda até os valores que possuem frequência >=5% e até o intervalo abranger, no mínimo, 50% das classes estudadas. Caso a frequência do valor médio seja inferior a 5%, a média passa a ser o ponto máximo e o deslocamento será feito apenas para a esquerda. A direção dos deslocamentos é conhecida, uma vez que as distribuições de frequência das medidas possuem uma cauda positiva direita. Então, quanto mais afastado do zero, menor será a frequência dos valores.

C2 – Deslocar, baseada na média, uma unidade em direção da moda até que o intervalo envolva, no mínimo, 50% das classes estudadas ou alcance o valor 0.

C3 – Deslocar, baseada na moda, uma unidade para a direita e esquerda, incluindo os valores de maiores frequência, até que o intervalo envolva, no mínimo, 50% das classes estudadas.

Na Tabela 44, apresentam-se os resultados obtidos após aplicar os três critérios definidos, em cada medida, nas 3 versões dos 107 sistemas de *software*. Na coluna um, são apresentadas as nomenclaturas das medidas; na coluna dois, é apresentado o rótulo das versões; na coluna três, é apresentado o rótulo dos critérios definidos; nas colunas quatro e cinco, são apresentados, respectivamente, os limites inferiores e superiores; na coluna seis, é apresentada a porcentagem de valores presentes no intervalo identificado.

Tabela 44 Aplicação dos critérios as medidas das versões 1, 2 e 3.

Medidas	Versão	Critério	LI	LS	% de classes no intervalo
WMC	V1	C1	2	7	65%
		C2	2	7	65%
		C3	2	4	50%
	V2	C1	2	7	65%
		C2	2	7	65%
		C3	2	4	50%
	V3	C1	2	7	65%
		C2	2	7	65%
		C3	2	5	56%
RFC	V1	C1	5	23	50%
		C2	5	23	50%
		C3	2	12	51%
	V2	C1	5	23	51%
		C2	5	23	51%
		C3	2	12	51%
	V3	C1	5	23	51%
		C2	5	23	51%
		C3	2	12	51%
CBO	V1	C1	2	7	52%
		C2	2	7	52%
		C3	1	5	57%
	V2	C1	2	7	52%
		C2	2	7	52%
		C3	1	5	54%
	V3	C1	2	7	53%
		C2	2	7	53%
		C3	1	5	54%
LCOM2	V1	C1	-	-	-
		C2	1	85	50%
		C3	0	1	53%
	V2	C1	-	-	-
		C2	1	100	51%
		C3	0	1	53%
	V3	C1	-	-	-
		C2	1	148	52%
		C3	0	1	54%
Ca	V1	C1	1	3	65%
		C2	1	3	65%
		C3	0	1	60%
	V2	C1	1	3	65%
		C2	1	3	65%
		C3	0	1	59%
	V3	C1	1	3	65%
		C2	1	3	65%
		C3	0	1	59%
Ce	V1	C1	1	5	52%
		C2	1	4	56%
		C3	0	3	58%
	V2	C1	1	5	61%
		C2	1	4	54%
		C3	0	3	55%
	V3	C1	1	6	65%
		C2	1	4	53%
		C3	0	3	55%

Na Tabela 44, é possível observar que os limites inferiores e superiores, da medida WMC, variam de [2, 7] nos critérios C1 e C2 das versões V1, V2 e V3 e abrangem, aproximadamente, 65% das classes dos sistemas de *software* estudados. Além disso, é possível observar que o limite inferior e superior no critério C3, das versões V1 e V2 variam de [2, 4] e abrangem, aproximadamente, 50% das classes dos sistemas de *software*. Por fim, verifica-se que, no critério C3 da versão V3, o limite inferior e superior varia de [2, 5] e abrange, aproximadamente, 56% das classes.

Na medida RFC apresentam-se limites inferiores e superiores que variam de [5, 23], nos critérios C1 e C2, das versões V1, V2 e V3 e abrangem, aproximadamente, 51% das classes. Também é possível observar, analisando a Tabela 44, que os limites inferiores e superiores, no critério C3, são iguais nas versões V1, V2 e V3 e variam de [2, 12].

Na medida CBO apresentam-se limites inferiores e superiores que variam de [2, 7], nos critérios C1 e C2, das versões V1, V2 e V3. Além disso, analisando a Tabela 44, verifica-se que os limites inferiores e superiores, no critério C3, são iguais nas versões V1, V2 e V3 e variam de [1, 5].

Na medida LCOM2, os critérios C1 e C2 mostraram-se ineficientes. A construção dos intervalos dos critérios C1 e C2 são baseadas na média global *Bootstrap*. Entretanto, em virtude da alta dispersão dos dados, as médias sofreram um grande deslocamento e inviabilizaram o uso dos critérios. O critério C3, na medida LCOM2, nas três versões estudadas, variou de [0, 1]. Tais valores abrangem, aproximadamente, 53% das classes dos sistemas de *software* da versão V1 e V2 e, aproximadamente, 54% das classes dos sistemas de *software* da versão V3.

Na medida Ca apresentam-se limites inferiores e superiores que variam de [1, 3] nos critérios C1 e C2 das versões V1, V2 e V3. Também, é possível

observar, analisando a Tabela 44, que os limites inferiores e superiores, no critério C3, são iguais nas versões V1, V2 e V3 e variam de [0,1].

Na medida Ce apresentaram-se limites inferiores e superiores que variam de [1, 5] no critério C1 nas versões V1 e V2 e de [1, 6] na versão V3. No critério C2 os limites inferiores e superiores variaram de [1, 4] nas três versões (V1, V2 e V3). Por fim, no critério C3 os intervalos variaram de [0, 3] nas três versões V1, V2 e V3, como apresentado na Tabela 44.

Na Tabela 45, são apresentados os resultados obtidos após aplicar os três critérios definidos nas medidas DIT e NOC. Na coluna um, são apresentadas as nomenclaturas das medidas; na coluna dois, é apresentado o rótulo das versões; na coluna três, é apresentado o rótulo dos critérios definidos; na coluna quatro, os valores mais frequentes; na coluna seis, é apresentada a porcentagem de classes presentes nos valores.

Tabela 45 *Bootstrap* aplicado a JHotDraw.

Medidas	Versão	Critério	Moda (Valor mais frequente)	Porcentagem de Classes no intervalo
DIT	V1	C1	1	51%
		C2	1	51%
		C3	1	51%
	V2	C1	1	52%
		C2	1	52%
		C3	1	52%
	V3	C1	1	53%
		C2	1	53%
		C3	1	53%
NOC	V1	C1	0	92%
		C2	0	92%
		C3	0	92%
	V2	C1	0	92%
		C2	0	92%
		C3	0	92%
	V3	C1	0	92%
		C2	0	92%
		C3	0	92%

Ao aplicar os critérios C1, C2 e C3 nos valores das versões V1, V2 e V3 das medidas DIT e NOC, foi possível observar que a média global *Bootstrap* é,

aproximadamente, igual à moda. Nas duas medidas, o valor mais frequente (Moda) abrange mais que 50% das classes estudadas. Na medida DIT, o valor mais frequente foi 1 que corresponde a mais de 51% dos valores das classes da versão V1, 52% dos valores das classes da versão V2 e 53% dos valores das classes da versão V3. Na medida NOC, o valor mais frequente foi 0 que corresponde a 92% dos valores das classes, nas três versões analisadas.

5.5.5 Análise de Correlação

Para verificar a existência de relacionamento, foi calculado o coeficiente de correlação de *Spearman* (R) entre os pares das medidas WMC, DIT, NOC, RFC, CBO, LCOM2, Ca, Ce e LOC. A escolha do coeficiente de correlação de *Spearman* foi em função da amostra coletada não obedecer à distribuição normal, necessitando de testes não paramétricos. O coeficiente de correlação de *Spearman* mede a intensidade da relação entre duas variáveis (SPEARMAN, 1904). Usa, em vez do valor observado, a ordem das observações. Desse modo, esse coeficiente é insensível a assimetrias, não exigindo que os dados sejam modelados por uma distribuição normal (SHARMA, 2005).

Ainda, o coeficiente de correlação *Spearman* varia de [-1, +1]. O sinal positivo e negativo (+,-) indicam a direção da relação. A relação positiva (+) indica que, quando uma variável independente cresce, a variável dependente, também, cresce. Ou o contrário, quando uma variável independente decresce, o valor, também, decresce. A relação negativa (-) indica que, quando as variáveis dependentes crescem, as variáveis independentes decrescem e vice-versa. Os valores extremos (-1 e +1) indicam uma relação perfeita e o 0 indica a ausência de correlação. Os valores extremos, dificilmente, são encontrados na prática. Quanto mais próximo dos extremos, maior a correlação. Quanto mais próximo

do zero, menor a correlação entre as variáveis (HEIMAN, 2011; SHARMA, 2005).

Estudos (COHEN, 1988; SUCCI et al., 2005) propõem a classificação dos coeficientes em: baixa, moderada e alta correlação. Portanto, para a interpretação da magnitude das correlações foi adotada a classificação dos coeficientes de correlação utilizada em Succi et al. (2005), que sugere:

(i) $r < 0.499$ (correlação baixa);

(ii) $0.500 \leq r < 0.799$ (correlação moderada);

(iii) $r > 0.800$ (correlação alta)

Foram adotados 5% como nível de significância estatística. Sendo o *p-value* calculado menor ou igual ao nível de significância predeterminado, a correlação entre as variáveis foi significativa. Adotou-se nível de significância de 5%, porque vem sendo empregado em outros estudos (EL EMAM; MELO; MACHADO, 2001; SUCCI et al., 2005) na área de Engenharia de *Software* e em diversas áreas.

Para todas as medidas foi calculado o coeficiente de determinação (R^2) e, para as medidas que apresentaram correlação alta, foi ajustada uma reta de regressão linear, que descreve a relação entre as medidas avaliadas. O coeficiente de determinação foi utilizado para definir o quanto uma medida pode explicar ou pode ser explicada pela outra.

Objetivando verificar a existência de relação entre as medidas coletadas, os coeficientes de correlação de *Spearman*, bem como o *p-value*, foram calculados, considerando todas as classes dos sistemas de *software*, separadamente para as três versões estudadas. Para o cálculo dos coeficientes de correlação de *Spearman* e *p-value*, foi utilizada a ferramenta IBM SPSS *Statistics* 20 (MAROCO, 2007). Os resultados das correlações podem ser visualizados nas Tabelas 46, 47 e 48.

Os coeficientes de correlação (R) assinalados com **, Tabelas 46, 47 e 48, indicam que as correlações são significativas ao nível de significância 1%. Os valores presentes na diagonal principal são sempre 1, pois se trata da correlação de uma medida com ela própria.

Analisando as Tabelas 46, 47 e 48, verifica-se a existência de correlação positiva alta, destacadas em negrito, entre as medidas RFC e WMC e entre as medidas RFC e LOC nas três versões analisadas. Também é possível observar a existência de correlações positivas moderadas, destacadas com sublinhado.

Na Tabela 46 apresentam-se os coeficientes de correlação e determinação dos sistemas de *software* da versão 1. Verifica-se que a medida WMC apresentou correlação positiva alta com a medida RFC (N=53.770; R = 0.826; *p-value* < 0.001). Conforme o valor da medida WMC aumenta, o valor da medida RFC, também, aumenta.

Tabela 46 Matriz de correlação não paramétrica das medidas selecionadas, considerando a versão 1 [N=53.770].

			WMC	**DIT**	**NOC**	**CBO**	**RFC**	**LCOM2**	**Ca**	**Ce**	**LOC**
Spearman's	WMC	R**	1								
		R²	100%								
	DIT	R**	-0.159	1							
		R²	3%	100%							
	NOC	R**	0.215	-0. 057	1						
		R²	5%	0%	100%						
	CBO	R**	0.544	-0.257	0.241	1					
		R²	30%	7%	6%	100%					
	RFC	R**	**0.826**	-0.12	0.162	0.588	1				
		R²	**68%**	1%	2%	35%	100%				
	LCOM	R**	0.749	-0.188	0.192	0.482	0.58	1			
		R²	56%	4%	4%	23%	34%	100%			
	CA	R**	0.277	0.128	0.301	0. 559	0.178	0.222	1		
		R²	8%	2%	9%	31%	3%	5%	100%		
	CE	R**	0.488	-0.34	0.102	0.738	0.663	0.406	0.064	1	
		R²	24%	12%	1%	54%	44%	16%	0%	100%	
	LOC	R**	0. 753	-0.115	0.147	0.516	**0.925**	0.485	0.144	0.596	1
		R²	57%	1%	2%	27%	**86%**	24%	2%	36%	100%

**. Correlações significativas a 1%.

Com o coeficiente de determinação (R^2 = 0.68), pode-se concluir que, aproximadamente, 68% da variação da medida WMC pode ser explicada pela variação da medida RFC, nos dados analisados.

Além da alta correlação com a medida WMC, a medida RFC apresentou alta correlação positiva com a medida LOC. À medida que o número de linhas aumenta a medida RFC, também, aumenta (N=53.770; R = 0.925; *p-value* < 0.001). Com o coeficiente de determinação (R^2 = 0.86), pode-se concluir que, aproximadamente, 86% da variação da medida WMC pode ser explicada pela variação da medida RFC, nos dados analisados.

Na Figura 5, são apresentados os *scatterplots* que ilustram as correlações positiva alta entre as medidas WMC e RFC e entre as medidas RFC e LOC.

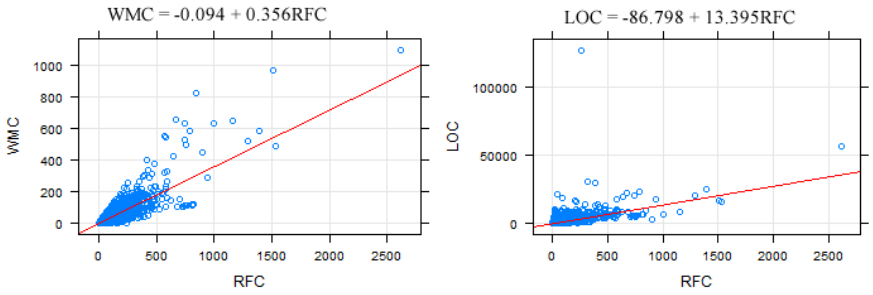

Figura 5 *Scatterplots* das correlações entre as medidas WMC e RFC (direita) e entre as medidas LOC e RFC (esquerda), considerando a versão 1.

Interpretando a Figura 5, tem-se que, para cada variação de uma unidade da medida RFC, a medida WMC cresce 0.356 unidades, como mostrado na equação presente na Figura 5 (direita). De forma similar, para cada variação de uma unidade da medida LOC, a medida RFC cresce 13.395 unidades, como mostrado na equação presente na Figura 5 (esquerda).

Tabela 47 Matriz de correlação não paramétrica das medidas selecionadas, considerando a versão 2 [N=77.513].

			WMC	DIT	NOC	CBO	RFC	LCOM2	Ca	Ce	LOC
Spearman's	WMC	R**	1								
		R²	100%								
	DIT	R**	-0.135	1							
		R²	2%	100%							
	NOC	R**	0.211	-0.054	1						
		R²	4%	0%	100%						
	CBO	R**	0.53	-0.244	0.232	1					
		R²	28%	6%	5%	100%					
	RFC	R**	**0.815**	-0.128	0.158	0.587	1				
		R²	**66%**	2%	2%	34%	100%				
	LCOM	R**	0.751	-0.159	0.187	0.463	0.565	1			
		R²	56%	3%	3%	21%	32%	100%			
	CA	R**	0.276	0.127	0.293	0.558	0.163	0.231	1		
		R²	8%	2%	9%	31%	3%	5%	100%		
	CE	R**	0.474	-0.341	0.098	0.736	0.674	0.378	0.055	1	
		R²	22%	12%	1%	54%	45%	14%	0%	100%	0%
	LOC	R**	0.733	-0.122	0.146	0.509	**0.921**	0.461	0.127	0.606	1
		R²	54%	1%	2%	26%	**85%**	21%	2%	37%	100%

**. Correlações significativas a 1%.

Na Tabela 47 apresentam-se os coeficientes de correlação e determinação dos sistemas de *software* da versão 2. Verifica-se que a medida WMC apresentou correlação positiva alta com a medida RFC (N=77.513; R = 0.815; *p-value* < 0.001). Conforme o valor da medida WMC aumenta, o valor da medida RFC, também, aumenta. Com o coeficiente de determinação (R^2 = 0.66), pode-se concluir que, aproximadamente, 66% da variação da medida WMC pode ser explicada pela variação da medida RFC, nos dados analisados.

Além da alta correlação com a medida WMC, a medida RFC apresentou alta correlação positiva com a medida LOC. À medida que o número de linhas aumenta a medida RFC, também, aumenta (N=77.513; R = 0.921; *p-value* < 0.001).

Com o coeficiente de determinação (R^2 = 0.85), pode-se concluir que, aproximadamente, 85% da variação da medida WMC pode ser explicada pela variação da medida RFC, nos dados analisados.

Na Figura 6 são apresentados os *scatterplots* que ilustram as correlações positiva alta entre as medidas WMC e RFC e entre as medidas RFC e LOC.

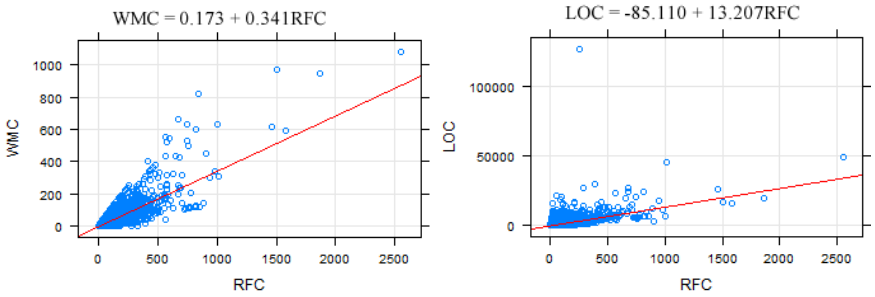

Figura 6 *Scatterplots* das correlações entre as medidas WMC e RFC (direita) e entre as medidas LOC e RFC (esquerda), considerando a versão 2.

Interpretando a Figura 6, tem-se que, para cada variação de uma unidade da medida RFC, a medida WMC cresce a uma taxa de 0.341, como mostrado na equação presente na Figura 6 (direita). De forma similar, para cada variação de uma unidade da medida RFC, a medida LOC cresce a uma taxa de 13.207, como mostrado na equação presente na Figura 6 (esquerda).

Na Tabela 48 apresentam-se os coeficientes de correlação e determinação dos sistemas de *software* da versão 3. Verifica-se que a medida WMC apresentou correlação positiva alta com a medida RFC (N=106.087; R = 0.822; *p-value* < 0.001). Conforme o valor da medida WMC aumenta, o valor da medida RFC, também, aumenta.

Com o coeficiente de determinação (R^2 = 0.68), pode-se concluir que, aproximadamente, 68% da variação da medida WMC pode ser explicada pela variação da medida RFC, nos dados analisados.

Tabela 48 Matriz de correlação não paramétrica das medidas selecionadas, considerando a versão 3 [N=106.087].

			WMC	DIT	NOC	CBO	RFC	LCOM2	Ca	Ce	LOC
Spearman's	WMC	R**	1								
		R²	100%								
	DIT	R**	-0.122	1							
		R²	1%	100%							
	NOC	R**	0.211	-0.051	1						
		R²	4%	0%	100%						
	CBO	R**	0.519	-0.228	0.223	1					
		R²	27%	5%	5%	100%					
	RFC	R**	**0.822**	-0.124	0.159	0.579	1				
		R²	**68%**	2%	3%	34%	100%				
	LCOM	R**	0.744	-0.148	0.19	0.449	0.558	1			
		R²	55%	2%	4%	20%	31%	100%			
	CA	R**	0.256	0.124	0.287	0.559	0.149	0.235	1		
		R²	7%	2%	8%	31%	2%	6%	100%		
	CE	R**	0.47	-0.334	0.095	0.735	0.67	0.359	0.054	1	
		R²	22%	11%	1%	54%	45%	13%	0%	100%	
	LOC	R**	0.738	-0.115	0.147	0.497	**0.92**	0.46	0.113	0.598	1
		R²	54%	1%	2%	25%	**85%**	21%	1%	36%	100%

**. Correlações significativas a 1%.

Além da alta correlação com a medida WMC, a medida RFC apresentou alta correlação positiva com a medida LOC. À medida que o número de linhas aumenta a medida RFC, também, aumenta (N=106.087; R = 0.920; *p-value* < 0.001).

Com o coeficiente de determinação ($R^2 = 0.85$), pode-se concluir que, aproximadamente, 85% da variação da medida WMC pode ser explicada pela variação da medida RFC, nos dados analisados.

Na Figura 7, são apresentados os *scatterplots* que ilustram as correlações positiva alta entre as medidas WMC e RFC e entre as medidas RFC e LOC.

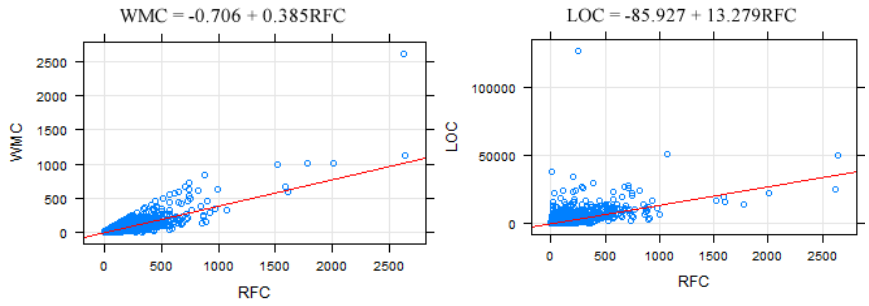

Figura 7 *Scatterplots* das correlações entre as medidas WMC e RFC (direita) e entre as medidas LOC e RFC (esquerda), considerando a versão 3.

Interpretando a Figura 7, tem-se que, para cada variação de uma unidade da medida RFC, a medida WMC cresce a uma taxa de 0.385, como mostrado na equação presente na Figura 7 (direita). De forma similar, para cada variação de uma unidade da medida RFC, a medida LOC cresce a uma taxa de 13.279, como mostrado na equação presente na Figura 7 (esquerda).

Verifica-se, analisando as Tabelas 46, 47 e 48, que as correlações entre as medidas das 3 versões estudadas, comportam-se de maneira similar. Nos três casos, pode-se notar a existência de uma correlação positiva alta entre as medidas WMC e RFC e entre as medidas LOC e RFC.

A correlação positiva alta entre as medidas WMC e RFC parece razoável, visto que à medida que WMC aumenta (ou seja, à medida que inserimos um novo método em uma classe) RFC, também, aumenta, pois será um novo método a ser executado em resposta a uma mensagem. De forma similar, o inverso, também, ocorre. Aumentando RFC, WMC, também, aumenta.

A correlação positiva alta entre as medidas LOC e RFC, também, parece razoável, pois à medida que o número de linhas (LOC) em uma classe aumenta a quantidade de métodos que podem ser invocadas em resposta a uma mensagem (RFC), também, aumenta.

Se WMC possui alta correlação positiva com a medida RFC e a medida RFC possui alta correlação positiva com a medida LOC, a transitividade pode ser verificada? Analisando as Tabelas 46, 47 e 48, verifica-se que as medidas WMC e LOC possuem correlação positiva moderada. A transitividade parece existir, porém não há evidencias suficiente para tal afirmação.

Ainda, analisando as Tabelas 46, 47 e 48, verifica-se que a medida Ca e Ce apresentaram correlações positivas moderada com a medida CBO nas três versões estudadas. A medida CBO utiliza a medida Ca e Ce em sua formula. Logo, conforme o valor das medidas Ca e/ou Ce aumentam o valor da medida CBO, também, aumenta. É interessante notar que a medida Ce apresentou um coeficiente de correlação mais elevado nas três versões analisadas, indicando que a medida Ce pode exercer mais impacto em CBO do que a medida Ca.

As medidas LCOM2 e WMC apresentaram uma correlação moderada positiva nas três versões. Logo, à medida que os valores de WMC aumentam os valores de LCOM, também, aumentam.

Nas Tabelas 46, 47 e 48, é possível observar que, das medidas estudadas, apenas DIT e NOC não apresentaram correlação moderada ou alta com nenhuma das medidas, considerando as amostras estudadas. Ainda, é possível observar que a medida DIT apresentou correlação baixa negativa com quase todas as medidas correlacionadas, exceto com a medida Ca a qual possui correlação baixa positiva. Uma correlação negativa sugere, que conforme o valor da medida DIT aumenta, os valores das demais medidas exceto Ca, diminuem. Entretanto, trata-se de correlações baixa.

Em suma, após a análise de correlação dos dados, conclui-se que as medidas estudadas estão correlacionadas. Entretanto, vale ressaltar que uma correlação não resulta em causa efeito. Duas medidas podem ter coeficientes de correlação altos e não haver relação de causa e efeito entre elas. Por exemplo, as medidas WMC e LOC possuem correlação positiva moderada, no entanto, não

significa que LOC elevado causa o aumento de WMC. Intuitivamente, o contrário teria uma relação de causa e efeito, pois à medida que WMC aumenta, em decorrência da inserção de novos métodos, LOC, também, aumenta.

Por fim, conclui-se que existe correlação entre as medidas estudadas, considerando o coeficiente de correlação de *Spearman*, para dados não paramétricos e significância de *p-value* < 0.001.

5.6 Lições Aprendidas

Durante a coleta das medidas, foram identificados fatores que impactam a medição e análise de medidas de *software*. Dentre eles: i) Definição das medidas; ii) Relatórios Complexos; iii) Definição do Artefato Analisado; iv) Performance; v) Indisponibilidade e Obsolescência; e vi) Ferramentas de Coleta. Vale ressaltar que os pontos listados foram baseados na experiência obtida ao longo da condução do projeto.

5.6.1 Definição das Medidas

Um dos principais problemas relacionados às medições de *software* é a falta de uma definição conceitual adequada das medidas. Existem diversas medidas que não possuem uma definição completa e adequada, fazendo com que sejam ambíguas. Por exemplo, verifica-se que a medida *Lines of Code* (LOC) contabiliza a quantidade de linhas em uma classe. Entretanto, não é definido se o número de comentários ou número de linhas em branco ou ambos são contabilizados na medida LOC. Outro exemplo é a medida WMC, presente na suíte CK (CHIDAMBER; KEMERER, 1994). Por definição, WMC é a soma dos métodos ponderados de uma classe. No entanto, método ponderado não é

definido em WMC. Essas ambiguidades ou falta de uma definição mais precisa impactam em seus cálculos.

Por causa da falta de definição adequada, involuntariamente, surgem variações e novas medidas com a mesma finalidade. Por exemplo, a medida LCOM, da suíte CK, possui diversas versões LCOM2, LCOM3, LCOM4, dentre outras. São tantas variações que se observa na falta de coesão que há na medida, porém não se nota como está sendo calculada, não se verifica a maneira adequada de interpretar o resultado, caso essas variações não sejam devidamente documentadas na ferramenta, que, geralmente, não acontece.

Verificou-se que algumas ferramentas implementam medida de *software* sem qualquer fundamentação teórica comprovada. Simplesmente, praticam e a disponibilizam, sem qualquer documentação que contribua para o entendimento do que é medido.

As medidas retornadas pelas ferramentas devem ser confiáveis, pois serão utilizadas por Engenheiros de *Software*, Arquitetos de *Software*, Gerente de Projetos como apoio na tomada de decisão. Para que sejam confiáveis, a ferramenta deve ser implementada de forma objetiva, as medidas devem ser implementadas conforme foram validadas, a documentação deve descrever adequadamente o que foi implementado.

5.6.2 Relatórios Complexos

Quando se trata de ferramentas de medição, um dos pontos críticos são seus relatórios. Verifica-se, analisando as ferramentas estudadas, que alguns relatórios são simples, com pouco ou nenhum recurso gráfico de visualização, apenas informam os valores das medidas para cada medida.

Constatou-se que não há preocupação, por parte de alguns desenvolvedores de ferramenta, com a interpretação das medidas calculadas.

Dentre as ferramentas estudadas, a que apresentou informações que auxiliam na interpretação das medidas provendo recursos, como: gráficos, descrições das medidas e fórmulas, foi a CodeProAnalityx. Algumas ferramentas, embora não forneçam um relatório que favoreça a interpretação, estruturam os dados de forma que possam ser importados por ferramentas que gerem gráficos.

Recomenda-se que as ferramentas de medição, além dos valores das medidas, tragam informações sugerindo como os valores poderiam ser interpretados, elementos visuais como gráficos e diagramas. Além disso, com base na medida, poderiam sugerir um diagnóstico para o *software* medido. Caso a medida exceda um valor determinado, poderiam ser sugeridas formas de como o valor da medida poderia ser melhorado.

É importante que os relatórios sejam de fácil entendimento, com recursos gráficos, sugestões de melhorias para que usuários menos familiarizados possam utilizar, sem que seja necessário esforço ou habilidades específicas.

5.6.3 Definição do Artefato Analisado

Dentre as ferramentas de medições analisadas, foram identificadas as que realizam cálculos com base no *bytecode* e código fonte. Entretanto, podem-se considerar, também, ferramentas para coleta de medidas em nível de projeto.

Análise de código fonte é uma técnica que extrai informações necessárias para os cálculos das medidas diretamente do código fonte. Atualmente, existem compiladores, como JavaCC ou ANTLR, que dão suporte à implementação de analisadores de código fonte. Entretanto, esses compiladores fornecem informações limitadas e, dependendo da profundidade e abrangência da medida, analisadores semânticos devem ser implementados manualmente.

Análise de *bytecode* é uma técnica que extrai as informações necessárias para os cálculos das medidas diretamente do código compilado. Atualmente, existem bibliotecas, como BCEL, SOOT e WALA, que fornecem informações extraídas diretamente do *bytecode*. Tais informações podem ser utilizadas para calcular medidas. Assim como a análise do código fonte, a análise de *bytecode* possui suas limitações. No momento que um código é compilado, são feitas otimizações no código original que podem interferir no cálculo das medidas.

Análise de projeto de *software* é uma técnica que extrai informações necessárias considerando um arquivo XMI, o qual contém um modelo de *software* em UML. Esse tipo de análise é importante, pois, ainda em nível de projeto, seria possível verificar medidas da arquitetura e sua complexidade.

Dependendo da característica do artefato de ser modelo, código fonte ou *bytecode*, podem existir valores distintos para as medidas. Por exemplo, há diversos recursos na programação Java, um deles é o *Innerclass* (classes que podem ser declaradas dentro de outras classes). Esse recurso é tratado de forma diferente nas ferramentas. As ferramentas que se baseiam em *bytecode*, aplicando a medida *Number of Files*, apresentaram valores superiores quando comparados com *software* que utilizam o código fonte. Isso ocorre, porque, no momento da compilação, o compilador Java cria um arquivo separado para a classe principal e para a *Innerclass*. Isso não ocorre para os sistemas de *software* baseados em código fonte. Se uma classe possui *Innerclass*, ela será mantida dentro do arquivo .java e, consequentemente, contabilizado por *Number of Files* apenas um arquivo.

Por isso, deve-se definir qual tipo de artefato será analisado e ter ciência que as comparações de medidas coletadas em tipos de artefatos distintos podem não ser diretamente comparáveis.

5.6.4 Performance

Durante o processo de medição dos sistemas de *software*, verificaram-se diferentes performances e consumos de memória para cada ferramenta utilizada. As coletas de medidas, de um modo geral, tiveram melhores desempenhos nas ferramentas que implementam a extração com base no *bytecode*. As ferramentas que fazem análise no código fonte, em alguns momentos, apresentaram problemas de travamento. Por exemplo, durante a seleção da ferramenta utilizada neste trabalho, diversas ferramentas foram testadas e a ferramenta Google CodePro, *plugin* do eclipse, em *software*, acima de 3.000 classes, apresentou problemas de travamento. Foi constatado que a ferramenta estava excedendo a memória da máquina e estava "paginando". Por causa disso, as análises, quando concluíam, demoravam consideravelmente.

5.6.5 Indisponibilidade e Obsolescência

Um dos grandes problemas enfrentados ao longo deste trabalho foi que muitas ferramentas eram citadas ou referenciadas na literatura, mas não estavam disponíveis. Possivelmente, elas foram desenvolvidas para fins acadêmicos específicos e não foram disponibilizadas. Pode-se observar que diversas ferramentas foram desenvolvidas porque as existentes não eram confiáveis, eficientes ou não apresentavam uma documentação adequada. Simplesmente, apresentam um número e não sugerem o que ele representa ou como ele foi obtido.

Outro problema encontrado, que inibe o uso de ferramentas e, consequentemente, a aplicação de medidas de *software* em empresas, é a obsolescência de algumas delas. Verifica-se que algumas ferramentas que, ainda, são utilizadas, deixaram de ser atualizadas há tempos. Como novas medidas e

interpretações são constantemente propostas e novos *bugs* em ferramentas são encontrados, as ferramentas devem estar em constante atualização, a fim de suprir as necessidades atuais.

5.6.6 Ferramentas de Coleta

Um dos principais desafios, ao longo deste trabalho, foi identificar uma ferramenta de coleta adequada, que forneça valores confiáveis. Foi verificado, nas ferramentas funcionais, inicialmente, encontradas, apresentadas na Tabela 46, que uma das principais causas de discrepância nos valores das medidas (resultados) entre as ferramentas é em razão da forma como o *input* é tratado.

Outros pontos devem ser listados, como: o cálculo das medidas se aplica a classes abstratas? A arquivos? Às interfaces? Classes anônimas? Em quais situações as interfaces são mensuradas? Na Tabela 49, é apresentado um quadro resumo de como o *input* é tratado por algumas ferramentas. Mas, vale ressaltar que, para cada medida, é possível encontrar uma particularidade. Por exemplo, a medida LOC, quando coletada em ferramentas que fazem análises, com base no *bytecode*, pode apresentar valores distintos. No *bytecode*, geralmente, os compiladores removem as linhas de comentário e linhas em branco, mas não há garantia que todos os compiladores façam isso. Além disso, quando um construtor não é declarado, ele é acrescido no *bytecode*. Assim, códigos compilados em compiladores diferentes podem apresentar valores diferentes porque não há um padrão para a criação de um *bytecode*. Outro exemplo é a inserção de um construtor, várias medidas podem ser afetadas, como: *Number of Methods, LCOM, Number of Public Methods*, LOC, dentre outras.

Tabela 49 Tratamento de input ferramentas de coleta de medidas.

Ferramentas	Classes	Files	Classes Abstratas	Interface	Classes Anônimas	InnerClass
Analizo.org	X		X	X		X
CodeAnalyzer		X				
CodePro Analytix		X				
Connecta	X	X	X	X	X	X
CKJM	X	X	X	X	X	X
CKJM Ext	X	X	X	X	X	X
Krakatau	X	X	X	X	X	X
SimpleCode Metrics	X					

6 ANÁLISE COMPARATIVA

Apresenta-se, neste Capítulo, uma análise comparativa entre os valores identificados na literatura com os valores praticados no mercado de *software* livre.

Os intervalos, para os valores praticados no mercado, apresentados nas tabelas que seguem, foram obtidos por análise dos gráficos de frequência absoluta (Apêndice F), aplicando-se os critérios C1, C2 e C3, descritos na Seção 5.5.4.2, nas medidas. Verificou-se que os intervalos identificados nas versões V1, V2 e V3 foram próximos, em alguns casos, iguais. Dessa forma, optou-se por comparar apenas os intervalos, obtidos com a versão 3, pois se trata de dados recentes, acima de 2010.

Nas Tabelas 8, 9, 10, 11, 12, 13 e 14 contêm a descrição RSL(LETRA) que indica que o valor de referência veio da RSL, cujo índice do artigo é a letra representada entre parênteses, seguida pela referência do artigo.

6.1 *Weight Methods per Class (WMC)*

Na Tabela 50, é apresentado o valor de referência da medida WMC, utilizado por Alan e Catal (2009), e os valores praticados no mercado de *software* livre, obtidos com a análise dos gráficos de frequência absoluta e estatística descritiva. São apresentados os intervalos obtidos com a aplicação dos critérios C1, C2 e C3 e o intervalo entre o Quartil 1 (Q1) e Quartil 3 (Q3), presentes na Tabela 38.

Verifica-se que os intervalos identificados com a aplicação dos critérios C1, C2 e C3, nos valores da versão 3 (V3) e o intervalo [Q1, Q3], encontram-se abaixo do valor da literatura. O valor de referência da medida WMC, utilizado

por Alan e Catal (2009), foi proposto com base em padrões da indústria e da experiência profissional. Sendo assim, tal valor tem sua validade questionada.

Tabela 50 Comparação entre os valores de referência sugeridos na literatura com os valores praticados no mercado, da medida WMC.

WMC				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
Literatura	A	Alan e Catal (2009)	14	Max
Práticas de Mercado (C1, C2)			[2, 7]	[Min, Max]
Práticas de Mercado (C3)			[2, 5]	[Min, Max]
Práticas de Mercado ([Q1, Q3])			[2, 8]	[Q1, Q3]

Conforme apresentado na Figura 8, o valor de referência definido por Alan e Catal (2009) pode definir um intervalo largo e classes, que devem ser revisadas porque fogem das práticas de mercado, são aceitáveis pelo valor de referência definido.

Observa-se que os intervalos definidos com os critérios C1 e C2 são iguais. O critério C3 foi estreito quando comparado aos critérios C1 e C2 e ao intervalo definido pelos quartis. Por fim, ao fazer uma intersecção dos intervalos, é possível observar que houve uma convergência no intervalo [2, 5] dos valores praticados no mercado de *software* livre com o valor proposto na literatura. Também é possível observar que a união, que varia de [0,14], pode definir um intervalo largo.

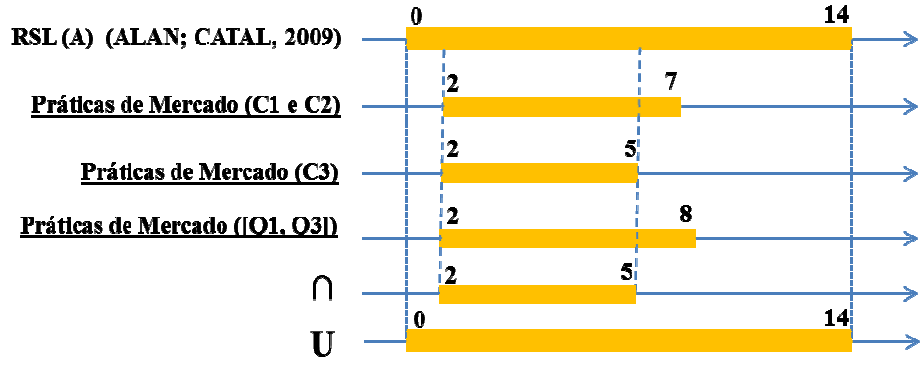

Figura 8 Representação dos intervalos da medida WMC.

6.2 *Depth of Inheritance Tree (DIT)*

Na Tabela 51, são apresentados os valores de referência da medida DIT, utilizados em Alan e Catal (2009), Benlarbi et al. (2000), Ferreira et al. (2009, 2011), Nair e Selvarani (2010) e Succi et al. (2005) e os valores praticados no mercado de *software* livre, obtidos com a análise dos gráficos de frequência absoluta e estatística descritiva. São apresentados os intervalos obtidos com a aplicação dos critérios C1, C2 e C3, descritos na seção 5.5.5 e o intervalo entre o Quartil 1 (Q1) e Quartil 3 (Q3), presentes na Tabela 38.

Verifica-se que os valores identificados após a aplicação dos critérios C1, C2 e C3 foi 1. O valor 1 representa o valor mais frequente da medida DIT, que corresponde a 53% dos valores praticados no mercado. O intervalo entre Q1 e Q3 abrange, aproximadamente, 86% dos valores praticados no mercado. Ou seja, a maioria dos valores da medida DIT está próxima de zero. Com esses valores é possível constatar que há pouco uso de Herança.

Tabela 51 Comparação entre os valores de referência sugeridos na literatura com os praticados no mercado, da medida DIT.

DIT				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
Literatura	Q	Succi et al. (2005)	10	Max
	A	Alan e Catal (2009)	7	Max
	D	Benlarbi et al. (2000)	6	Max
	Q	Succi et al. (2005)	6	Max
	S	Nair e Selvarani (2010)	3 e 6	Desejável e Max
	K	Ferreira et al. (2009)	2	Típico
	L	Ferreira et al. (2011)	2	Típico
Práticas de Mercado (C1, C2 e C3)			1	Moda
Práticas de Mercado ([Q1, Q3])			[0, 1]	[Q1, Q3]

Verifica-se que os valores mais frequentes das práticas do mercado de *software* livre aproximam-se dos valores de referência propostos por Ferreira et al. (2009, 2011), Nair e Selvarani (2010) e Succi et al. (2005). Em Ferreira et al. (2009, 2011), foi definido DIT=2 como valor típico. Tal valor parece razoável, uma vez que utilizam diversos sistemas de *software opensource* e análises estatísticas. Em Nair e Selvarani (2010), DIT=3 é sugerido como um valor desejável, entretanto, a técnica empregada para obtenção do valor foi a experiência profissional e o valor não foi validado empiricamente, logo, tem sua validade questionada.

Os valores definidos em Alan e Catal (2009) e Succi et al. (2005) parecem representar um intervalo largo. É importante ressaltar que DIT elevado é indesejado, porque quanto maior a profundidade na árvore de herança, maior a dificuldade de prever o comportamento das classes herdeiras, consequentemente, maior a complexidade do sistema.

Observa-se que os intervalos definidos com os critérios C1, C2 e C3 foram iguais e representam o valor mais frequente. Por fim, ao fazer uma intersecção dos intervalos, é possível observar que não houve convergência dos valores praticados no mercado de *software* livre com os valores propostos na literatura. Também é possível observar que a união, que varia de [0,10], pode definir um intervalo largo.

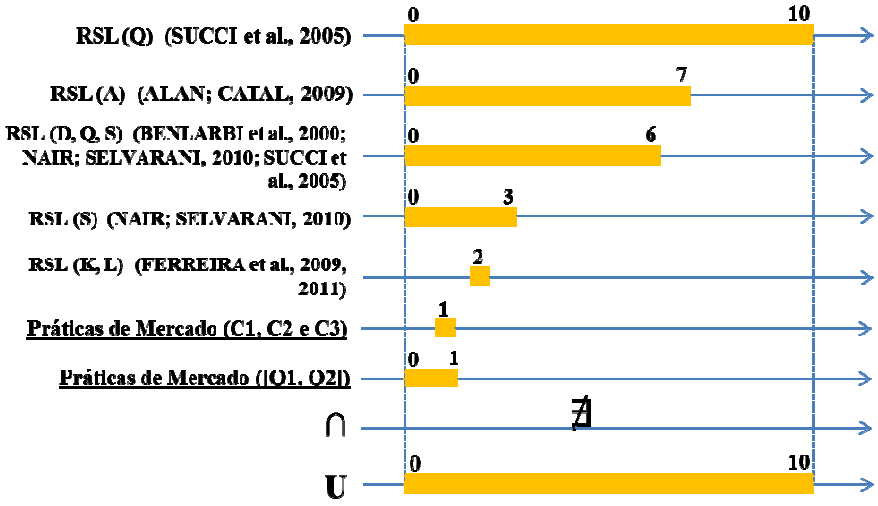

Figura 9 Representação dos intervalos da medida DIT.

6.3 *Number of Children (NOC)*

Na Tabela 52, é apresentado o valor de referência da medida NOC, utilizado em Alan e Catal (2009) e Succi et al. (2005), e os valores praticados no mercado de *software* livre, obtidos com a estatística descritiva e análise dos gráficos de frequência absoluta.

Analisando a Tabela 52, é possível observar que o valor mais frequente nas práticas de mercado de *software* livre, da medida NOC, é 0. Esse valor

abrange 92% dos valores estudados. O valor NOC = 0 indica pouco uso de classes filhas. Observa-se que os valores de referência sugeridos em Alan e Catal (2009) e Succi et al. (2005) são divergentes dos valores praticados no mercado.

Tabela 52 Comparação entre os valores de referência sugeridos na literatura com os praticados no mercado, da medida NOC.

	NOC			
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
Literatura	Q	Succi et al. (2005)	10	Max
	Q	Succi et al. (2005)	4	Max
	A	Alan e Catal (2009)	3	Max
Práticas de Mercado (C1, C2 e C3)			0	Moda
Práticas de Mercado ([Q1, Q3])			[0, 0]	[Q1, Q3]

Os valores citados por Succi et al. (2005), NOC = 10, pode ser considerado um intervalo largo, pois é elevado quando comparado aos demais valores apresentados. Na Figura 10, é apresentada uma representação dos valores identificados na literatura e das práticas de mercado.

Observa-se que os intervalos definidos com os critérios C1, C2 e C3 foram iguais e representam o valor mais frequente. Por fim, ao fazer uma intersecção dos intervalos, é possível observar que, no 0, houve convergência dos valores praticados no mercado com os valores propostos na literatura. Também é possível observar que a união, que varia de [0,10], pode definir um intervalo largo.

A medida NOC, segundo Chidamber e Kemerer (1994), é um indicador de reutilização e quanto maior for o número de filhos de uma classe, maior sua reutilização. Entretanto, quanto maior o número de filhos, mais difícil será de

modificá-la, pois as modificações afetam todos os seus filhos, que têm uma forte dependência com a classe mãe.

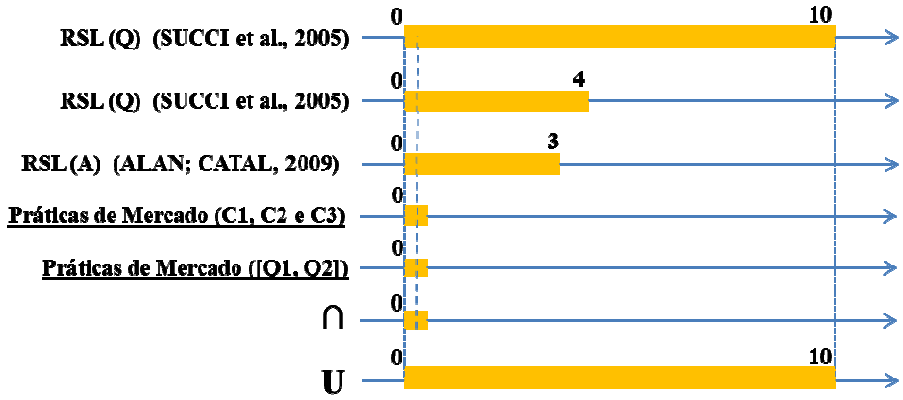

Figura 10 Representação dos intervalos da medida NOC.

6.4 *Response For Class (RFC)*

Na Tabela 53, são apresentados os valores de referência da medida RFC, utilizados em Alan e Catal (2009), Benlarbi et al. (2000), Herbold, Grabowski e Waack (2011), Nair e Selvarani (2010), Shatnawi (2010) e Shatnawi et al. (2010), os valores praticados no mercado de *software* livre, obtidos com a estatística descritiva e análise dos gráficos de frequência. São apresentados os intervalos obtidos com a aplicação dos critérios C1, C2 e C3 e o intervalo entre o Quartil 1 (Q1) e Quartil 3 (Q3).

Verifica-se que os intervalos identificados, após a aplicação dos critérios C1, C2 e C3, nos valores da Versão 3 e o intervalo [Q1, Q3], obtido com a estatística descritiva, encontram-se abaixo dos valores de referência presentes na literatura.

Tabela 53 Comparação entre os valores de referência sugeridos na literatura com os valores praticados no mercado, da medida RFC.

RFC				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
Literatura	A	Alan e Catal (2009)	100	Max
	D	Benlarbi et al. (2000)	100	Max
	J	Shatnawi (2010)	100	Max
	J	Shatnawi (2010)	40	Max
	O	Shatnawi et al. (2010)	100	Max
	O	Shatnawi et al. (2010)	44	Max
	R	Herbold, Grabowski e Waack (2011)	100	Max
	R	Herbold, Grabowski e Waack (2011)	98	Max
	S	Nair e Selvarani (2010)	[50, 100] e 222	Desejável e Max
Práticas de Mercado (C1, C2)			[5, 23]	[Min, Max]
Práticas de Mercado (C3)			[2, 12]	[Min, Max]
Práticas de Mercado ([Q1, Q3])			[5, 25]	[Q1, Q3]

Comparando com os valores propostos na literatura, o que mais se aproximou dos valores de prática de mercado foi o valor de referência proposto por Shatnawi (2010), RFC = 40. Entretanto, a discrepância entre os valores é alta. Assim, os valores de referência propostos por Shatnawi (2010) e demais autores podem representar intervalos largos e considerar classes que devem ser revisadas como aceitáveis.

Verifica-se, analisando a Figura 11, que os valores de referência sugeridos e utilizados na literatura não convergem com os valores das práticas de mercado de *software* livre. Além disso, é possível observar que a união, que varia de [0, 222], pode definir um intervalo largo.

Intervalos de referência largos podem não ser interessantes para a medida RFC, visto que RFC é diretamente proporcional à complexidade geral

do sistema. Dessa forma, quanto maior o valor da medida RFC, maior é a complexidade geral do sistema.

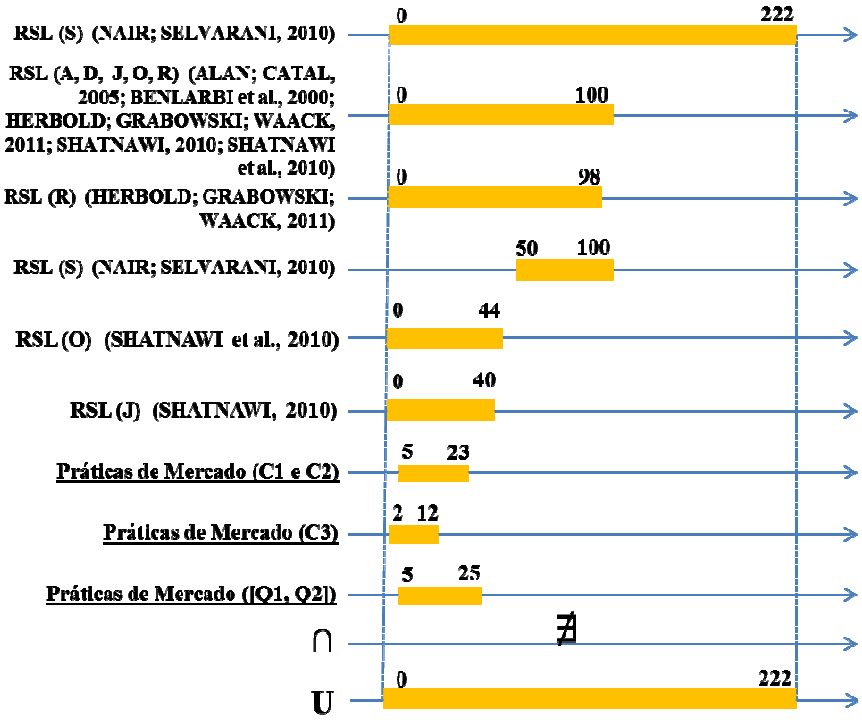

Figura 11 Representação dos intervalos da medida RFC.

6.5 *Coupling Between Object Classes (CBO)*

Na Tabela 54, são apresentados os valores de referência da medida CBO, utilizados em Alan e Catal (2009), Benlarbi et al. (2000), Herbold, Grabowski e Waack (2011), Shatnawi (2010) e Shatnawi et al. (2010), os valores praticados no mercado, obtidos com a estatística descritiva e análise dos gráficos de frequência absoluta. São apresentados os intervalos obtidos com a

aplicação dos critérios C1, C2 e C3 e o intervalo entre o Quartil 1 (Q1) e Quartil 3 (Q3).

Verifica-se que os intervalos identificados após a aplicação dos critérios C1, C2 e C3, nos valores da versão 3 (V3) e o intervalo [Q1, Q3], obtido com a estatística descritiva, encontram-se abaixo do valor de referência (CBO = 13) utilizado em Shatnawi et al. (2010).

Tabela 54 Comparação entre os valores de referência sugeridos na literatura com os valores praticados no mercado, da medida CBO.

CBO				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
Literatura	A	Alan e Catal (2009)	2	Max
	D	Benlarbi et al. (2000)	5	Max
	J	Shatnawi (2010)	5	Max
	J	Shatnawi (2010)	9	Max
	O	Shatnawi et al. (2010)	5	Max
	O	Shatnawi et al. (2010)	13	Max
	R	Herbold, Grabowski e Waack (2011)	5	Max
Práticas de Mercado (C1, C2)			[2, 7]	[Min, Max]
Práticas de Mercado (C3)			[1, 5]	[Min, Max]
Práticas de Mercado ([Q1, Q3])			[2, 9]	[Q1, Q3]

Dentre as técnicas utilizadas para definir valores de referência, tanto os valores definidos por meio da experiência profissional quanto às demais técnicas, parecem definir um intervalo razoável. Na Figura 12, são representados os intervalos dos valores de referência identificados na literatura e os intervalos dos valores praticados no mercado. Observa-se que os valores de referência propostos na literatura estão próximos dos valores praticados no mercado.

Verifica-se que os intervalos definidos com os critérios C1 e C2, são iguais. A amplitude do intervalo obtido pelo critério C3 foi estreito, quando comparado aos critérios C1 e C2 e com o intervalo definido pelos quartis. Por fim, ao fazer uma intersecção dos intervalos, é possível observar que houve uma

convergência no valor 2, dos valores praticados no mercado de *software* livre com os valores propostos na literatura. É importante mencionar que a convergência entre os valores foi igual ao valor mais frequente da medida CBO na versão 3, como apresentado na Tabela 38. Também é possível observar que a união, que varia de [0,13] pode definir um intervalo largo.

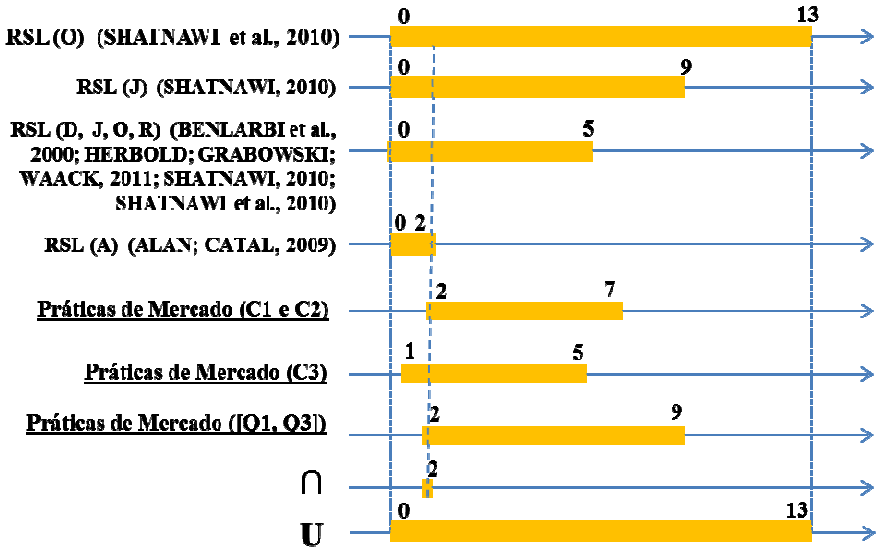

Figura 12 Representação dos intervalos da medida CBO.

6.6 *Lack of Cohesion Metric (LCOM 2)*

Na Tabela 55, são apresentados os valores de referência da medida LCOM2, utilizados em Ferreira et al. (2009, 2011) e Shatnawi et al. (2010), e os valores praticados no mercado de *software* livre, obtidos com a análise dos gráficos de frequência absoluta e estatística descritiva. São apresentados os

intervalos obtidos com a aplicação dos critérios C3 e o intervalo entre o Quartil 1 (Q1) e Quartil 3 (Q3), presentes na Tabela 38.

Os critérios C1 e C2 mostraram-se ineficientes na medida LCOM2 por serem baseados na média. Como a dispersão dos dados foi alta, a média sofreu um grande deslocamento e impactou nos intervalos gerados pelo critério C1 e C2. Dessa forma, os intervalos C1 e C2 não foram utilizados na comparação, apenas o critério C3 e o intervalo entre Q1 e Q3.

Tabela 55 Comparação entre os valores de referência sugeridos na literatura com os valores praticados no mercado, da medida LCOM2.

LCOM2				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
Literatura	K	Ferreira et al. (2009)	0,]0;20[e >=20	Bom, Regular e Ruim
	L	Ferreira et al. (2011)	0, [1;20] e >20	Bom, Regular e Ruim
	O	Shatnawi et al. (2010)	8	**3º** Quartil
Práticas de Mercado (C3)			[0, 1]	[Min, Max]
Práticas de Mercado ([Q1, Q3])			[0, 12]	[Q1, Q3]

Na Figura 13, é apresentada uma representação dos intervalos da medida LCOM2. Analisando o intervalo [Q1, Q3], é possível notar que 75% dos valores da medida LCOM2 são menores ou iguais a 13. Esse valor é maior que o valor do terceiro quartil definido em Shatnawi et al. (2010). Entretanto, é importante ressaltar que os valores definidos por Shatnawi et al. (2010) foram estabelecidos com base em quatro sistemas, enquanto os valores apresentados neste trabalho são baseados em 3 versões de 107 sistemas de *software* livres distintos. Ainda, é possível observar que o valor do terceiro quartil está entre os valores regulares definidos em Ferreira et al. (2009, 2011).

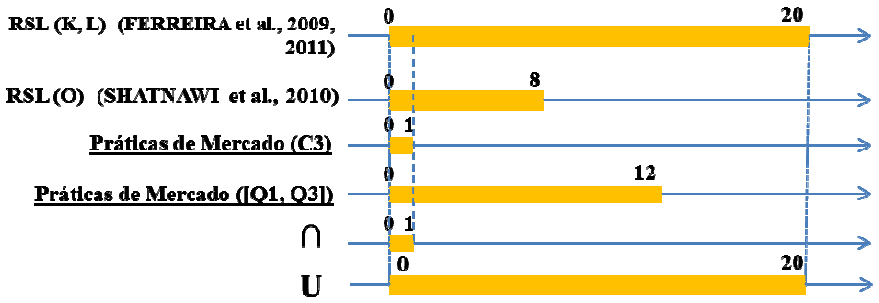

Figura 13 Representação dos intervalos da medida LCOM2.

Observa-se que o intervalo definido com o critério C3 foi estreito e variou de [0, 1]. Porém, esse intervalo é significativo, pois abrange 54% dos valores analisados. Também, é interessante notar que o valor 0 é o mais frequente, com mais de 40% dos valores analisados. Por fim, ao fazer uma intersecção dos intervalos, é possível observar que houve uma convergência no intervalo [0, 1] dos valores praticados no mercado com o valores propostos na literatura. Também é possível verificar que a união, que varia de [0,20], pode definir um intervalo largo.

6.7 *Afferent Coupling (Ca)*

Na Tabela 56, são apresentados os valores de referência da medida CA, utilizados em Alves, Ypma e Visser (2010) e Ferreira et al. (2009, 2011), e os valores praticados no mercado, obtidos com a estatística descritiva e análise dos gráficos de frequência absoluta.

Constata-se, analisando a Figura 14, que os valores de prática do mercado de *software* livre estão distantes dos valores propostos na literatura. Por exemplo, em Alves, Ypma e Visser (2010) são definidos os valores dos percentis para as medidas. 80% dos valores são menores ou iguais a 22.

Tabela 56 Comparação entre os valores de referência sugeridos na literatura com os praticados no mercado, da medida Ca.

Ca e FAN-IN				
	Índice	**Ref.**	**Valor**	**Natureza da Medida**
Literatura	B	Alves, Ypma e Visser (2010)	10,22 e 56	70%, 80%, 90% percentils
	K	Ferreira et al. (2009)	0,]1;20[e >=20	Bom, Regular e Ruim
	L	Ferreira et al. (2011)	1, [2;20] e >20	Bom, Regular e Ruim
Práticas de Mercado (C1, C2)			[1, 3]	[Min, Max]
Práticas de Mercado (C3)			[0, 1]	[Min, Max]
Práticas de Mercado ([Q1, Q3])			[1, 3]	[Q1, Q3]

O terceiro quartil dos valores praticados no mercado foi 3. Por fim, ao fazer uma intersecção dos intervalos, é possível ressaltar que houve convergência, no valor Ca = 1, entre os valores praticados no mercado com os valores propostos na literatura. Também é possível observar que a união, que varia de [0, 56] pode definir um intervalo largo.

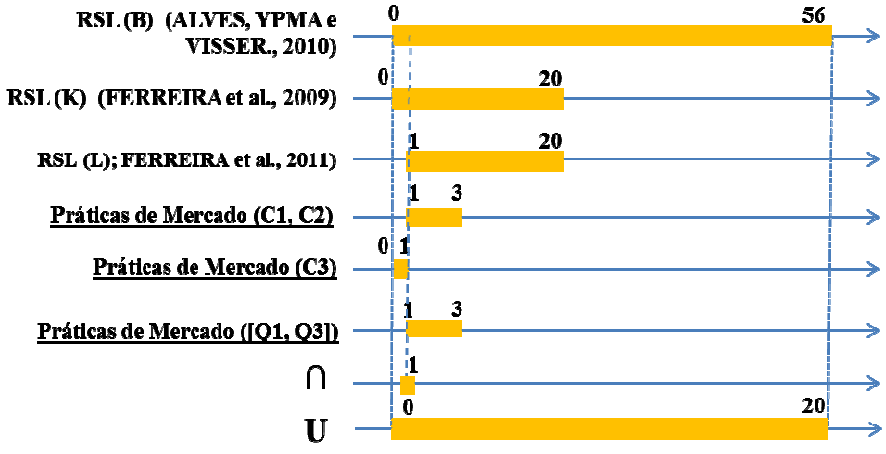

Figura 14 Representação dos intervalos das medidas AC e *Fan-in.*

6.9 *Efferent Coupling (Ce)*

Infelizmente, nenhum valor que pudesse ser comparado foi encontrado para a medida Ce. Entretanto, os valores identificados neste trabalho representam a prática de mercado de *software* livre e podem ser adotados como referência.

Ao aplicar o critério C1, foi identificado o intervalo [1, 6] que abrange, aproximadamente, 65% dos valores analisados. Ao aplicar o critério C2, foi identificado um intervalo [1, 4] que abrange, aproximadamente, 53% dos valores analisados. Ao aplicar o critério C3, o intervalo [0, 3] que abrange 55% dos valores estudados. Conforme a Tabela 38, o intervalo [Q1, Q3] varia de [1, 6]. Fazendo a intersecção dos valores, define-se um intervalo [1, 3]. Fazendo a união, tem-se um intervalo que varia de [0, 6].

6.10 Considerações Finais

A tarefa de estabelecer valores de referência não é trivial. Dentre algumas alternativas, tomando a medicina como exemplo, poder-se-ia pegar indivíduos saudáveis que, após questionário, apresentam-se “saudáveis” e “normais” e, considerando uma amostra representativa, estabelecer os valores mínimos e máximos extraídos com base na amostra.

Extrapolar essa técnica para a Engenharia de *Software* é possível e demanda alto custo, como na medicina. Para determinar quais sistemas de *software* são “saudáveis”, seria necessário que vários especialistas fizessem uma inspeção no projeto e/ou código do sistema, a fim de obter a amostra dos sistemas de *software* “saudáveis”. Logo depois, calcular os intervalos das medidas.

Neste trabalho, limitou-se a comparar os valores de referência sugeridos na literatura e os valores praticados no mercado de *software* livre. Observa-se que dependendo da medida, a discrepância entre os valores variam, mas é possível a

comunidade científica verificar o que é praticado no mercado, bem como o mercado observar alguns valores preconizados pela academia.

Após a análise, conclui-se que os valores de propostos na literatura e os valores praticados no mercado são convergentes nas medidas WMC, NOC, CBO, Ca e LCOM2 e divergentes nas medidas DIT e RFC. De modo geral, os valores de referência, propostos na literatura, definem intervalos largos, quando comparados com os intervalos identificados na prática de mercado.

Nenhum valor foi identificado na literatura para a medida Ce. Dessa forma, os intervalos obtidos com os valores praticados no mercado podem ser utilizados como referência, uma vez que representam mais de 50% dos valores presentes em 107 sistemas de *software* distintos.

Os valores de DIT e NOC são inferiores aos definidos na literatura, entretanto, os valores aqui definidos representam sistemas de *software* reais e utilizados no mercado. Os valores baixos podem ser um indicativo de que os recursos da programação orientada a objetos estejam sendo pouco utilizados pelos desenvolvedores ou tais recursos estejam sendo evitados.

Dentre as medidas selecionadas, as medidas LOC e nCL não foram comparadas uma vez que não possuem limite natural. Os valores são variáveis conforme a característica, porte, tecnologia de cada sistema.

Os valores praticados no mercado podem servir para que as empresas possam ter um intervalo para comparar as medidas de seus sistemas de *software* com as medidas dos sistemas de *software* disponíveis no mercado. Assim, é possível, de maneira preliminar, comparar a “qualidade” do que se prática pelo mercado e a “qualidade” produzida por uma empresa.

Um intervalo delimita uma região de interesse ou um conjunto de intervalos mais confiáveis (SCHWAAB; PINTO, 2007). Neste trabalho, foram definidos, com base em três critérios, intervalos para os valores mais usuais e praticados pelo mercado de sistemas de *software opensource*. Entretanto, valores

abaixo ou acima do intervalo são valores que podem ocorrer, mas são casos especiais, incomuns ou até mesmo indesejados. Valores externos aos intervalos indicam mudança de comportamento dos valores observados ou aparecimento de um fato novo, até então desconsiderado. Sendo assim, recomenda-se que se avalie o código que apresentar valor fora dos intervalos calculados, a fim de garantir que não há erro de projeto ou programação.

As medidas de *software* vêm exercendo um papel fundamental nas organizações. No entanto, embora tenha havido um crescimento nos últimos 10 anos de estudos, relacionados a medidas de *software* e valores de referência, esse assunto, ainda, carece de pesquisas e publicações que forneçam suporte a engenheiros de *software* ou interessados.

Ao longo deste trabalho, foram verificados diversos fatores que impactam diretamente na aceitação das medidas. Dentre eles, está a falta de definição adequada das medidas que, de um modo geral, possuem descrições ambíguas e causam diversas interpretações. Outro fator são os relatórios complexos que proveem pouca facilidade para usuários que não possuem conhecimento técnico. A falta de valores de referência, objeto de estudo desse trabalho, também, faz com que as medidas sejam pouco utilizadas nas empresas, dentre outros fatores.

É importante ressaltar que os intervalos não definem níveis de qualidade, ou seja, não quer dizer que os valores entre os intervalos definidos representam as melhores práticas da Engenharia de *Software*. Os valores representam um intervalo de confiança obtido com base em sistemas de código aberto desenvolvidos em Java. Logo, os valores aqui encontrados ficam restritos a tal linguagem. Estudos adicionais são necessários para aprofundar a questão e ser possível, no futuro, estabelecer medidas com valores de referência confiáveis e bem definidos, a fim de apoiar os Engenheiros de *Software* na construção e diagnóstico de *software*.

7 CONCLUSÕES

O objetivo principal, na condução deste trabalho, foi realizar uma análise comparativa entre os valores de referência das principais medidas de *software*, apresentadas na literatura científica, e os valores praticados atualmente pelos desenvolvedores de *software*. Para isso, foi feito o levantamento das medidas de *software* que possuem valores de referência associados a ela, por meio de uma Revisão Sistemática da Literatura. Posteriormente, foram realizadas análises estatísticas em 10 medidas de *software*. Por fim, compararam-se os valores das medidas identificadas na RSL com os valores das medidas coletados no mercado.

Durante a condução da RSL, foram identificados 4.735 artigos após buscas nas bases: *IEEE Xplore, Ei Compendex, Elsevier Science Direct, Scopus e ACM Library*. Dos 4.735 artigos científicos, 184 foram selecionados na etapa de seleção primária, dos quais foram lidos o Título e as Palavras-chave. Dentre os 184 artigos, 137 foram considerados irrelevantes, 28 foram considerados repetidos e nenhum foi considerado incompleto, restando 19 artigos que foram lidos e analisados no decorrer da RSL.

As questões originais que motivaram essa RSL foram: a) quais medidas de *software* possuem valores de referência atribuídos a elas? b) Quais são os valores identificados na literatura? Ambas as questões foram respondidas e detalhes foram discutidos ao longo deste trabalho. No entanto, durante as análises, ficou evidente que os valores, ainda, não são suficientemente genéricos ou suficientemente validados. Além de conflitos entre alguns valores sugeridos.

Pôde-se observar na RSL que uma série de estudos foram realizados com base em valores de referência estabelecidos por meio da experiência profissional. Contudo, valores de referência, obtidos com base em experiência profissional podem não ser confiáveis. Logo, artigos que os utilizam para a elaboração de modelos ou diagnósticos podem ter seus resultados comprometidos.

Neste trabalho, também, foram apresentados os resultados de uma coleta e análise de dados referente à grande quantidade de sistemas de *software opensourse* desenvolvidos em Java, que representam as práticas de mercado. Foram analisadas 3 versões de 107 sistemas de *software*, desenvolvidos em Java, subdivididos em 25 domínios de aplicação, totalizando 237.370 classes e 321 versões de *software*.

Para cada medida selecionada, foi realizada uma análise preliminar dos dados para verificar a normalidade e identificar os testes apropriados. Foi constatado que os dados violam os pressupostos de análises paramétricas, dessa forma, foram adotados testes não paramétricos para a análise dos dados. Posteriormente, foram geradas estatísticas descritivas, para resumir os dados coletados e apresentar valores descritivos, como variância, desvio padrão, moda, mediana, média, dentre outros. Para cada medida, também, foram identificados os intervalos sem *outliers*, com *outliers* suaves e *outliers* extremos.

Foi realizado um estudo para averiguar a dinâmica entre as três versões dos sistemas de *software* estudados. Nesse estudo, foram verificadas 7 situações, dentre elas, crescimento médio contínuo e decrescimento médio contínuo das medidas selecionadas.

Foi realizado o teste de *Kruskall-Wallis*, com significância estatística de 5%, para verificar a existência de diferença significativa entre as três versões de um mesmo *software*. Posteriormente, foi aplicado o teste post-hoc de comparações múltiplas para comparar todos os sistemas de *software* entre si.

Foi aplicada a técnica *Bootstrap* e, com isso, foram calculadas médias *Bootstrap.* Em seguida, foram elaborados três critérios para definir os intervalos das práticas de mercado de *software* livre.

Na análise de correlação, foi calculado o coeficiente de correlação de *Spearman* e significância entre cada par de medidas selecionadas. Nessa análise, constatou-se a existência de uma correlação positiva alta entre as medidas WMC

e RFC e entre as medidas RFC e LOC. Também, foi constatada a correlação positiva baixa, próximo de zero, entre as medidas NOC e CE e entre as medidas NOC e DIT.

Por fim, foi apresentada uma análise comparativa entre os valores de referência propostos na literatura com os valores praticados no mercado de software livre. Na comparação, são apresentadas as medidas que divergiram e quais convergiram, após comparar os valores de referência.

O cenário atual conduz a comunidade de Engenharia de *Software* a aplicar mais esforços, para estabelecer os valores de referência. Desenvolver ferramentas de coleta que forneçam relatórios com recursos visuais e diagnósticos, com sugestões de melhorias, documentar, adequadamente, as medidas propostas, para que a mesma não venha a ter dupla interpretação e sejam erroneamente implementadas em ferramentas de coleta.

7.1 Contribuições

Dentre as principais contribuições deste trabalho, destacam-se:

1) a identificação de um conjunto de medidas com valores de referência propostos na literatura;
2) a categorização das medidas quanto ao contexto, à técnica e à validação;
3) avaliação crítica de 19 artigos científicos;
4) análise das medidas praticada no mercado de *software* livre;
5) elaboração de quadros comparativos entre as medidas identificadas;
6) identificação de correlação entre as medidas;

7) mapeamento da variação de medidas entre as versões do mesmo *software*;

8) identificação de intervalos de confiança, por meio da técnica de reamostragem *Bootstrap* e análise de gráficos de frequência;

9) identificação de diferenças significativas entre versões de um mesmo *software*.

7.2 Limitações

As possíveis ameaças à validade deste estudo, em relação à RSL, estão nas limitações das máquinas de busca, que, em geral, não estão preparadas a receber uma RSL. Outros fatores são: a qualidade dos estudos, caso não possuam um título, palavras-chave ou resumo escritos adequadamente, trabalhos relevantes podem ter sido excluídos pelos autores no momento das filtragens ocorridas com a execução da RSL.

Em relação aos testes realizados, está no uso de técnicas não paramétricas. Ressalta-se que técnicas não paramétricas não são tão robustas quanto as técnicas paramétricas. Entretanto, foi verificado que as pressuposições dos testes paramétricos foram violadas, restando, assim, os testes não paramétricos. Houve uma investigação, para identificar a melhor distribuição que modela os dados, foi utilizada a ferramenta EasyFit, para fazer o melhor ajuste, mas, foi constatado que a ferramenta não funciona de forma adequada em um grande volume de dados. Nos testes de hipótese realizados pela ferramenta, o melhor ajuste realizado é rejeitado. Dessa forma, foram utilizadas técnicas não paramétricas porque não é necessário fazer suposições das distribuições ou fazer considerações subjetivas.

Em relação aos valores comparados, é difícil afirmar se os valores de referência, propostos na literatura, foram calculados, adequadamente, uma vez que as informações contidas nos artigos são limitadas e, muitas vezes, insuficientes para fazer qualquer afirmação ou análise. Entretanto, tais artigos sugerem valores que podem ser utilizados como referência. Neste trabalho, foram identificados intervalos com base em análise de 3 versões de 107 sistemas de *software*. Por meio da comparação, é possível o mercado ter uma ideia do que vem acontecendo na academia e a academia ter ideia do que acontece no mercado.

7.3 Trabalhos Futuros

Várias ações podem ser tomadas como trabalhos futuros, e algumas delas já estão sendo realizadas pelo grupo de pesquisa:

1) realizar uma análise aprofundada das diferentes técnicas para calcular valores de referência, encontrados na literatura;
2) aplicar as técnicas de cálculo de valores de referência na base de dados utilizada neste trabalho, para fins comparativo com os intervalos obtidos;
3) refazer as análises utilizando outras medidas de *software*;
4) fazer análise comparativa das ferramentas de medição de *software* identificando discrepâncias de medições e as razões para tais discrepâncias;
5) análise da correlação entre medidas similares como, por exemplo, entre medidas de complexidade;

6) fazer revisões sistemáticas, isoladamente, para cada medida, a fim de obter mais detalhes ou novos valores que possam contribuir com a busca pelos valores de referência;

7) desenvolver e propor um protocolo para facilitar a pesquisa de valores de referência para várias métricas e tipos de *software*, e para instâncias de *software* específicos para assegurar o compartilhamento de dados e replicabilidade;

8) estabelecer protocolos de qualidade, por exemplo, estabelecer quais medidas são boas para mensurar manutenibilidade e criar técnicas bem definidas com medidas e valores de referência para servir de guia aos engenheiros de *software*.

REFERÊNCIAS

ABREU, F. B. E.; CARAPUÇA, R. Object-oriented *software* engineering: measuring. In: INTERNATIONAL CONFERENCE ON *SOFTWARE* QUALITY, 4., 1994, McLean. **Proceedings…** Milwaukee: American Society for Quality, 1994. p. 1-8.

ALAN, O.; CATAL, C. An outlier detection algorithm based on object-oriented metrics thresholds. In: INTERNATIONAL SYMPOSIUM ON COMPUTER AND INFORMATION SCIENCES, 24., 2009, Guzelyurt. **Proceedings…** Guzelyurt: IEEE Computer Society, 2009. p. 567-570.

ALBRECHT, A. J. Measuring application development productivity. In: IBM APPLICATION DEVELOPMENT SYMPOSIUM, 1., 1979, New York. **Proceedings…** New York: IBM, 1979. p. 83-92.

ALVES, T. L.; CORREIA, J. P.; VISSER, J. Benchmark-based aggregation of metrics to ratings. In: INTERNATIONAL WORKSHOP ON *SOFTWARE* MEASUREMENT, 21.; INTERNATIONAL CONFERENCE ON *SOFTWARE* PROCESS AND PRODUCT MEASUREMENT, 6., 2011, Nara. **Proceedings…** Nara: IEEE Computer Society, 2011. p. 20-29.

ALVES, T. L.; YPMA, C.; VISSER, J. Deriving metric thresholds from benchmark data. In: IEEE INTERNATIONAL CONFERENCE ON *SOFTWARE* MAINTENANCE, 10., 2010, Timisoara. **Proceedings…** Timisoara: IEEE Computer Society, 2010. p. 1-10.

BANSIYA, J.; DAVIS, C. A hierarchical model for object-oriented design quality assessment. **IEEE Transactions on *Software* Engineering**, Piscataway, v. 28, n. 1, p. 4-17, Jan. 2002.

BENLARBI, S. et al. Thresholds for object-oriented measures. In: INTERNATIONAL SYMPOSIUM ON *SOFTWARE* RELIABILITY ENGINEERING, 11., 2000, San Jose. **Proceedings…** San Jose: IEEE Computer Society, 2000. p. 24-38.

BIEMAN, J.; KANG, B. Cohesion and reuse in an object-oriented system. In: SYMPOSIUM ON *SOFTWARE* REUSABILITY, 1., 1995, Seattle. **Proceedings…** Seattle: ACM, 1995. p. 259-262.

BIOLCHINI, J. et al. **Systematic review in *software* engineering**. Rio de Janeiro: UFRJ, 2005. 31 p. (Technical Reports RT - ES, 679/05).

BRIAND, L. C.; WUST, J. The impact of design properties on development cost in object-oriented systems. In: INTERNATIONAL *SOFTWARE* METRICS SYMPOSIUM, 17., 2001, London. **Proceedings…** London: IEEE Computer Society, 2001. p. 260-271.

CATAL, C.; ALAN, O.; BALKAN, K. Class noise detection based on *software* metrics and ROC curves. **Information Sciences**, New York, v. 181, n. 21, p. 4867-4877, Nov. 2011.

CATAL, C.; SEVIM, U.; DIRI, B. Clustering and metrics thresholds based *software* fault prediction of unlabeled program modules. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY NEW GENERATIONS, 6., 2009, Las Vegas. **Proceedings…** Las Vegas: IEEE Computer Society, 2009. p. 199-204.

CCCC - C and C++ code counter. Disponivel em: <http://cccc.sourceforge.net/>. Acesso em: 15 ago. 2012.

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

CHIDAMBER, S. R.; KEMERER, C. F. Towards a metric suite for object oriented design. In: CONFERENCE OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES, AND APPLICATIONS, 6., 1991, New York. **Proceedings...** New York: ACM, 1991. p. 197-211.

CODEANALYZER. **CodeAnalyzer:** multi-platform java code analyzer. Disponivel em: <http://www.codeanalyzer.teel.ws/>. Acesso em: 21 maio 2012.

COHEN, J. **Statistical power analysis for the behavioral sciences**. 2nd ed. New Jersey: L. Erlbaum, 1988. v. 1, 590 p.

COPELAND, T. **PMD applied:** an easy-to-use guide for developers. Alexandria: Centennial Books, 2005. 224 p.

COPPICK, J. C.; CHEATHAM, T. J. *Software* metrics for object-oriented systems. In: ACM COMPUTER SCIENCE CONFERENCE, 20., 1992, New York. **Proceedings…** New York: ACM, 1992. p. 317-322.

DALLAL, J. A. Fault prediction and the discriminative powers of connectivity-based object-oriented class cohesion metrics. **Information and *Software* Technology**, New York, v. 54, n. 4, p. 396-416, Apr. 2012.

DALLAL, J. A. Improving the applicability of object-oriented class cohesion metrics. **Information and *Software* Technology**, New York, v. 53, n. 9, p. 914-928, Mar. 2011.

DALLAL, J. A.; BRIAND, L. C. A precise method-method interaction-based cohesion metric for object-oriented classes. **ACM Transactions on *Software* Engineering and Methodology**, New York, v. 21, n. 2, p. 1-34, Mar. 2012.

DYBÅ, T.; DINGSØYR, T.; HANSSEN, G. K. Applying systematic reviews to diverse study types: an experience report. In: INTERNATIONAL SYMPOSIUM ON EMPIRICAL *SOFTWARE* ENGINEERING AND MEASUREMENT, 1., 2007, Washington. **Proceedings…** Washington: IEEE Computer Society, 2007. p. 225-234.

EFRON, B.; TIBSHIRANI, R. J. **An introduction to the bootstrap**. New York: Chapman & Hall, 1993. 456 p.

EL EMAM, K. et al. The optimal class size for object-oriented *software*. **IEEE Transactions on *Software* Engineering**, New Jersey, v. 28, n. 5, p. 494-509, Aug. 2002.

EL EMAM, K.; MELO, W.; MACHADO, J. C. The prediction of faulty classes using object-oriented metrics. **Journal of Systems and *Software***, London, v. 56, n. 1, p. 63-75, Feb. 2001.

FENTON, N. E.; NEIL, M. *Software* metrics: successes, failures and new directions. **Journal of Systems and *Software***, London, v. 47, n. 2/3, p. 149-157, June 1999.

FENTON, N. E.; PFLEEGER, S. ***Software* metrics:** a rigorous and practical approach. Boston: PWS, 1997. 638 p.

FERREIRA, D. F. **Estatística computacional em Java**. Lavras: UFLA, 2013. v. 1, 695 p.

FERREIRA, K. A. M. et al. Identifying thresholds for object-oriented *software* metrics. **The Journal of Systems and *Software***, New York, v. 85, n. 2, p. 244-257, June 2011.

FERREIRA, K. A. M. et al. Reference values for object-oriented *software* metrics. In: BRAZILIAN SYMPOSIUM ON *SOFTWARE* ENGINEERING, 23., 2009, Fortaleza. **Proceedings…** Fortaleza: IEEE Computer Society, 2009. p. 62--72.

FRENCH, V. A. Establishing *software* metric thresholds. In: INTERNATIONAL WORKSHOP ON *SOFTWARE* MEASUREMENT, 9., 1999, Mont-Tremblant. **Proceedings…** Mont-Tremblant, 1999. p. 1-10.

GOOGLE CODEPRO ANALYTIX. **Java developer tools google codepro analytix:** Google developers. Disponivel em: <https://developers.google.com/java-dev-tools/codepro/doc/features/metrics/metrics?hl=pt-BR>. Acesso em: 20 maio 2012.

GUI, G.; SCOTT, P. D. New coupling and cohesion metrics for evaluation of *software* component reusability. In: INTERNATIONAL CONFERENCE FOR YOUNG COMPUTER SCIENTISTS, 9., 2008, Hunan. **Proceedings…** Hunan: IEEE Computer Society, 2008. p. 1181-1186.

HALSTEAD, M. H. **Elements of *software* science**. New York: Elsevier Science, 1977. 142 p.

HEIMAN, G. W. **Basic statistics for the behavioral sciences**. 6th ed. Belmont: Wadsworth Cengage Learning, 2011. 504 p.

HENRY, S.; KAFURA, D. *Software* structure metrics based on information flow. **IEEE Transactions on *Software* Engineering**, New York, v. SE-7, n. 5, p. 510-518, Sept. 1981.

HERBOLD, S.; GRABOWSKI, J.; WAACK, S. Calculation and optimization of thresholds for sets of *software* metrics. **Empirical *Software* Engineering**, Berlin, v. 16, n. 6, p. 812-841, Dec. 2011.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **Std 610.12:** standard glossary of *software* engineering terminology. New York, 1990. Disponivel em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnu>. Acesso em: 5 jan. 2012.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 9126:** information technology: *software* product evaluation: quality characteristcs and guidelines for their use. Geneva, 1991. 13 p.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 14598:** information technology: *software* product evaluation. Geneva, 1999. 14 p.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 25000:** *software* engineering: *software* product quality requirements and evaluation (SQuaRE). Geneva, 2014. 27 p.

JABREF. **Jabref s*oftware***. Disponivel em: <http://jabref.sourceforge.net/index.php>. Acesso em: 25 abr. 2012.

JURECZKO, M.; SPINELLIS, D. Using object-oriented design metrics to predict *software* defects. In: INTERNATIONAL CONFERENCE ON DEPENDABILITY OF COMPUTER SYSTEMS, 5., 2010, Wroclaw. **Proceedings…** Wroclaw: Oficyna Wydawnicza Politechniki Wroclawskiej, 2010. p. 69-81.

KAN, S. H. **Metrics and models in *software* quality engineering**. 2nd ed. Boston: A. Wesley, 2002. 560 p.

KIRBY, K. N.; GERLANC, D. **BootES:** an R package for bootstrap confidence intervals on effect sizes. 4th ed. Berlin: Springer, 2013. 927 p.

KITCHENHAM, B. **Procedures for performing systematic reviews**. Keele: Keele University, 2004. 33 p. (Technical Report TR/SE-0401; NICTA Technical Report, 0400011T.1).

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in *software* engineering**. Keele: EBSE Technical Report, 2007. 57 p. (EBSE-2007-01).

KRUSKALL, W. L.; WALLIS, D. H. Use of ranks in one-criterion variance analysis. **Journal of the American Statistical Associates**, New York, v. 47, n. 260, p. 583-621, 1952.

LANZA, M.; MARINESCU, R. **Object-oriented metrics in practice:** using *software* metrics to characterize, evaluate, and improve the design of object-oriented systems. New York: Springer, 2006. 205 p.

LINCKE, R.; LUNDBERG, J.; LÖWE, W. Comparing *software* metrics tools. In: INTERNATIONAL SYMPOSIUM ON *SOFTWARE* TESTING AND ANALYSIS, 8., 2008, Seattle. **Proceedings…** Seattle: ACM, 2008. p. 131-141.

LORENZ, M.; KIDD, J. **Object-oriented *software* metrics**. Upper Saddle River: Prentice-Hal, 1994. 146 p.

MAFRA, S. N.; BARCELOS, R. F.; TRAVASSOS, G. L. H. Aplicando uma metodologia baseada em evidência na definição de novas tecnologias de *software*. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE *SOFTWARE*, 20., 2006, Florianópolis. **Anais...** Florianópolis: SBC, 2006. p. 239-254.

MAROCO, J. **Análise estatística com utilização do SPSS**. 3. ed. Lisboa: Edições Sílabo, 2007. v. 1, 824 p.

MARTIN, R. OO design quality metrics: an analysis of dependencies. **ROAD**, Ave, v. 2, n. 3, p. 1-8, Sept./Oct. 1995.

MCCABE IQ. **MaCabe *software***. Disponivel em: <http://www.mccabe.com/pdf/McCabe%20IQ%20Metrics.pdf>. Acesso em: 22 maio 2012.

MCCABE SOFTWARE. **McCabe IQ glossary of terms:** McCabe *software*. Disponivel em: <http://www.mccabe.com/iq_research_iqgloss.htm>. Acesso em: 22 maio 2012.

MCCABE, T. J. A Complexity measure. **IEEE Transactions on *Software* Engineering**, New York, v. SE-2, n. 4, p. 308-320, Dec. 1976.

NAIR, T. R. G.; SELVARANI, R. Estimation of *software* reusability: an engineering approach. **ACM SIGSOFT *Software* Engineering Notes**, New York, v. 35, n. 1, p. 1-6, Jan. 2010.

POWERSOFTWARE. **Krakatau metrics professional**. Disponivel em: <http://www.power*software*.com/>. Acesso em: 10 maio 2012.

PRECHELT, L.; UNGER, B. An experimental measuring the effects of Personal *Software* Process (PSP) training. **IEEE Transactions on *Software* Engineering**, New Jarsey, v. 27, n. 5, p. 465-472, May 2001.

PRESSMAN, R. S. **Engenharia de *software*:** uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011. 780 p.

R CORE TEAM. **R:** a language and environment for statistical computing. Vienna: R Foundation for Statistical Computing, 2014. 3604 p.

RAMLER, R.; WOLFMAIER, K.; NATSCHLAGER, T. Observing distributions in size metrics: experience from analyzing large *software* systems. In: ANNUAL INTERNATIONAL COMPUTER *SOFTWARE* AND APPLICATIONS CONFERENCE, 31., 2007, Beijing. **Proceedings…** Beijing: IEEE Computer Society, 2007. p. 299-304.

RANDOLPH, J. J. A guide to writing the dissertation literature review. **Practical Assessment, Research & Evaluation**, Washington, v. 14, n. 13, p. 1-13, June 2009.

ROSENBERG, L. H. Applying and interpreting object oriented metrics. In: *SOFTWARE* TECHNOLOGY CONFERENCE, 10., 1998, Utah. **Proceedings…** Utah, 1998. p. 1-18.

ROSENBERG, L. H.; HYATT, L. ***Software* quality metrics for object-oriented system environments**. Washington: NASA Technical Report SATC, 2001. 58 p.

ROSENBERG, L. H.; STAPKO, R.; GALLO, A. Risk-based object oriented testing. In: ANNUAL *SOFTWARE* ENGINEERING WORKSHOP, 24., 1999, Greenbelt. **Proceedings…** Greenbelt: NASA, 1999. 1 CD-ROM.

SCHNEIDEWIND, N. F. *Software* metrics model for quality control. In: INTERNATIONAL *SOFTWARE* METRICS SYMPOSIUM, 4., 1997, Albuquerque. **Proceedings...** Albuquerque: IEEE Computer Society, 1997. p. 127-136.

SCHWAAB, M.; PINTO, J. C. **Análise de dados experimentais I:** fundamentos de estatística e estimação de parâmetros. Rio de Janeiro: E-Papers, 2007. v. 1, 462 p.

SHARMA, A. K. **Text book of correlations and regression**. New Delhi: Discovery, 2005. 212 p.

SHATNAWI, R. A quantitative investigation of the acceptable risk levels of object-oriented metrics in open-source systems. **IEEE Transactions on *Software* Engineering**, New York, v. 36, n. 2, p. 216-225, Mar. 2010.

SHATNAWI, R. et al. Finding *software* metrics threshold values using ROC curves. **Journal of *Software* Maintenance and Evolution: Research and Practice**, New York, v. 22, n. 1, p. 1-16, Jan. 2010.

SIMPLECODEMETRICS. **Simple code metrics:** plugin. Disponivel em: <http://plugins.netbeans.org/plugin/9494/simple-code-metrics>. Acesso em: 25 maio 2012.

SKYLAR, L.; SMITH, M. R. Evaluation of several nonparametric bootstrap methods to estimate confidence intervals for *software* metrics. **IEEE Transactions on *Software* Engineering**, New York, v. 29, n. 11, p. 996-1004, Nov. 2003.

SOMMERVILLE, I. ***Software*** **engineering**. 9th ed. Boston: A. Wesley, 2011. 773 p.

SOURCEFORGE. **Find, create, and publish open source *software* for free**. Disponivel em: <http://sourceforge.net/>. Acesso em: 3 Aug. 2012.

SPEARMAN, C. General intelligence, objectively determined and measured. **American Journal of Psychology**, Champaing, v. 15, n. 2, p. 201-293, 1904.

SPINELLIS, D. Tool writing: a forgotten art? **IEEE *Software***, New York, v. 22, n. 4, p. 9-11, Aug. 2005.

SUCCI, G. et al. An empirical exploration of the distributions of the Chidamber and Kemerer object-oriented metrics suite. **Empirical *Software* Engineering**, Berlin, v. 10, n. 1, p. 81-103, Jan. 2005.

TEMPERO, E. On measuring Java *software*. In: AUSTRALASIAN COMPUTER SCIENCE CONFERENCE, 31., 2008, Wollongong. **Proceedings…** Wollongong: CRPIT, 2008. p. 7-7.

TERCEIRO, A. et al. Analizo: an extensible multi-language source code analysis and visualization toolkit. In: BRAZILIAN CONFERENCE ON *SOFTWARE*, 1., 2010, Salvador. **Anais...** Salvador: SBC, 2010. p. 1-6.

TRIOLA, M. F. **Introdução à estatística**. 9. ed. Rio de Janeiro: LTC, 2005. v. 1, 682 p.

WASHIZAKI, H.; YAMAMOTO, H.; FUKAZAWA, Y. A metrics suite for measuring reusability of *software* components. In: INTERNATIONAL SYMPOSIUM ON *SOFTWARE* METRICS, 9., 2003, Sydney. **Proceedings…** Sydney: IEEE Computer Society, 2003. p. 211-223.

WOLFMAIER, K.; RAMLER, R. Common findings and lessons learned from *software* architecture and design analysis. In: IEEE INTERNATIONAL *SOFTWARE* METRICS SYMPOSIUM, 11., 2005, Como. **Proceedings…** Como: IEEE Computer Society, 2005. p. 1-8.

APÊNDICE A – INFORMAÇÕES DOS VALORES DE REFERÊNCIA DA LITERATURA.

Tabela 57 Detalhamento das informações dos valores de referência identificados na literatura.

Índice	**Ano**	***Software* usado para definir os valores de referência**	***Software* usado para validação**	**Domínio**	**Tamanho**
A	2009	-	-	-	-
B	2010	100 Sistemas OO (Java) e C#, *opensource*, desenvolvidos em diferentes organizações.	-	100 *Software*, classificados em 19 domínios: Catálogo ou registro de coisas ou eventos - 8 Faturamento ou gestão de relacionamento - 5 Gestão de documentos- 5 Transferência de dados eletrônicos - 3 Processamento de transações financeiras e de Contabilidade - 12 Sistemas de informações geográficas - 2 Gráficos e ferramentas de publicação - 2 *Software* integrado para controle da máquina - 3 Gestão de projetos - 6 Logística ou fornecimento de planejamento e Controle - 8 Gestão de desempenho ou relatórios - 2 Modelagem matemática - 1 Análise on-line e relatórios - 6 Sistemas operacionais ou utilitário de *software* - 14 Ferramenta de desenvolvimento de *Software* - 3 Controle de estoque e processamento de pedido - 1 Trading - 1 Suporte e gerenciamento de fluxo de trabalho - 10 Outros - 8	Total de 12 Milhões de linhas LOC
C	2011	-	ArgoUML usado como amostra, não para validar	1 – *Software* de Desenvolvimento	128.316 LOC

Tabela 57, continua

D	2000	Utilizou dados de outros artigos	2 sistemas em C++	Telecomunicações	Sistema 1 - 85 classes Sistema 2 - 174 classes
E	2002	Utilizou dados de outros artigos	2 sistemas em C++	Telecomunicações	Sistema 1 - 83 classes Sistema 2 - 174 classes
F	2009	3	Não validado	Controle de *software* White-bens	Desconhecido
G	2003	125 javabeans obtidos em www.jars.com	1 *software* FukaGraphBean em 3 versões desenvolvidas pelo propio autor..	125 *software*, classificados em 5 domínios de aplicação: Programação - 98 WWW - 4 Game - 3 Utilitários - 4 Ciência - 16	A média do numbers of properties é 5.14, e a média do numbers of business methods é 8.23
H	2007	6 projetos diferentes e 6 versões consecutivas de um único sistema para analisar a arquitetura e design. Em Java e C + +	Não validado	Diferentes domínios, não especificados.	De 60KLOC a 2.000 KLOC

Tabela 57, continua

I	1997	1397 módulos usados para extração de dados e validação	100 módulos	Space Shuttle AirCraft Flight *Software*	1397 módulos
J	2010	2 *software* (Eclipse project Version 2 and 2.1). Size V2 is 4.454 classes and V2.1 is 5.259 classes	2 (Sistemas Mozilla e Rhino)	Para extração dos dados (2 Desenvolvimento), para validação (1 Browser, 1 JavaScript Engine)	--
K	2009	40 *software*, *opensource*, desenvolvidos em Java.	Não validado	Foram utilizados 11 domínios: Clustering - 5 Banco de Dados - 3 Desktop - 5 Desenvolvimento - 7 Negócio - 4 Financeiro - 1 Jogos - 3 Hardware - 3 Multimedia 3 Rede 4 Segurança 2	26.202 classes. Detalhes podem ser visto no artigo original
L	2011	40 *software*, *opensource*, desenvolvidos em Java.	Experimento I - Algumas classes de nove sistemas. (Jasper Reports, KolfMafia, JavaX11Library, Hibernate, jpilot, CodeGeneration, BCEL, YAWL, Super) Experimento II - JHotDraw (1.095 classes avaliadas)	Foram utilizados 11 domínios: Clustering - 5 Banco de Dados - 3 Desktop - 5 Desenvolvimento - 7 Negócio - 4 Financeiro - 1 Jogos - 3 Hardware - 3 Multimedia 3 Rede 4 Segurança 2	Mais de 26.000 classes. Detalhes podem ser visto no artigo original

Tabela 57, conclusão

M	2011	-	4 *software* Java Illusion V2.5; GanttProject V2.0.5; JabRef V2.3 Beta 2; Openbravo V0.0.24.	Modelagem 3D, renderização e animação estúdio Cronograma do Projeto Gestão de bases de dados bibliográficas TouchScreen Hardware	Illusion 481 classes, cerca de 88Kloc GanttProject 468 classes, cerca de 39 Kloc JabRef 569 classes, cerca de 48KLOC Openbravo 447 classes, cerca de 36 KLOC
N	2011	5 sistemas da NASA (Conjunto de dados chamado: CM1, JM1, KC1, KC2, PC1) 2 sistemas em C, 3 sistemas in C++	--	1 Domínio. Predição de defeitos em *software*.	--
O	2010	1 *software*. (Versões 2.0, 2.1, 3.0 do eclipse).	--	Desenvolvimento	--
P	1992	1	--	1 Dominio (Graphics. É um *software* simples que envolve retângulo, círculo e painéis)	8 classes
Q	2005	100 C++ e 100 Java	--	Foi utilizados domínios diferentes, porém, não foram especificados.	Média number of classes in C++ is 59 Média number of classes in Java is 83,5
R	2011	--	--	--	--
S	2010	2 projetos java	--	--	Projeto 1 consiste em 265 classes das quais 96 são reutilizáveis. Projeto II consiste em 423 classes das quais 198 são reutilizáveis.

APÊNDICE B – MEDIDAS CONFORME OS ARTIGOS DA RSL.

Quadro 4 Descrição das medidas identificadas na literatura.

NOME	SIGLA	DESCRIÇÃO
Weight Methods per Class	WMC	Duas definições foram encontradas para a medida WMC, aqui chamadas de WMC1 e WMC2: - WMC1 – Contagem de métodos - WMC2 – Soma das complexidades dos métodos.
Depth of Inheritance Tree	DIT	O comprimento máximo do nó até a raiz da árvore de herança.
Number of Children	NOC	Número de subclasses imediatas subordinas a uma classe na hierarquia
Coupling between Object Classes	CBO	Contabiliza o número de classes às quais uma determinada classe está acoplada
Response for a Class	RFC	Corresponde ao número de métodos que podem ser executados em resposta a uma mensagem recebida por um objeto da classe.
Coupling Factor	COF	Se a classe A tem pelo menos uma referência para a classe B, que são acoplados, não importa como referências manu eles têm, apenas contar uma. Assim, em um sistema de nível, somar cada classe conectada a outras classes.
Fan-in / Fan-out	-	Indica o número de métodos ou funções que chamam outros métodos ou funções / Indica o número de métodos ou funções chamadas pelo método ou função.
Afferent / Efferent Coupling	AC / EC	Fornece o número total de classes externas a um pacote que dependem de classes de dentro desse pacote / Fornece o número total de classes dentro de um pacote que dependem de classes externas ao pacote.
Cyclomatic Complexity per Class / per Method/ per Function	CC	Calcula o número de caminhos linearmente independentes através de um grafo de fluxo.

Quadro 4, continua.

Line of Code	LOC	Esta medida apresenta diferentes interpretações, como: - Número de linhas de código, sem espaço em branco e comentários; - Número de linhas de código, com espaço em branco, sem comentários; - Número de linhas de código, sem espaço em branco, com linhas de comentários; - Número de linhas de código, com espaço em branco e linhas de comentário; - Número de linhas de declarações; - Número de linhas de quantidade de retorno.
Lack of Cohesion of Methods	LCOM	Foram encontradas, na literatura, diferentes definições para essa medida. Neste trabalho, foi adotado: LCOM1, LCOM2, LCOM3, LCOM4, LCOM5 e LOCM: - LCOM1 - Conta o número de pares de métodos que não compartilham variáveis de instância - LCOM2 - Calcula a diferença entre o número de pares de métodos que não compartilham variáveis de instância; - LCOM3-Um grafo não direcionado, que representa cada método como um nó e a partilha de uma variável de instância como uma vantagem. É medido em termos de número de componentes conectados no gráfico. - LCOM4 - é LCOM3 adicionado de uma borda de um par de métodos quando um invoca o outro. - LCOM5 - é dada por (a-kl) / (l-kl), onde **l** é o número de atributos, **k** é o número de métodos e **a** é a soma de o número de atributos distintos acessado por cada um dos métodos de uma classe. - LOCM - é a média em todos os domínios de uma classe do percentual dos métodos de classe "que usam esse campo, com o resultado subtraído de 100 Definido em Ferramentas McCabe.

APÊNDICE C – MEDIDAS CONFORME A FERRAMENTA CKJM EXT.

Quadro 5 Descrição das medidas calculadas por CKJM Ext.

Nome	Sigla	Descrição
Weight Methods per Class	WMC	Atribui valor 1, a cada método, e posteriormente faz um somatório desses valores para cada classe, logo, WMC coletado corresponde ao número de métodos em uma classe.
Depth of Inheritance Tree	DIT	Calcula, para cada classe, o número máximo de superclasses posicionadas hierarquicamente acima.
Number of Children	NOC	Calcula o número de descendentes imediatos de cada classe;
Coupling Between Object	CBO	Calcula o número de classes acopladas a uma determinada classe (acoplamentos eferentes e acoplamentos aferentes). O acoplamento pode ocorrer através de chamadas de método, campo de acessos, a herança, argumentos, tipos de retorno e exceções.
Response for Class	RFC	Calcula o número de métodos diferentes que podem ser executado quando um objeto dessa classe recebe uma mensagem (quando um método é invocado para esse objeto). Assim, RFC foi calculado somando os métodos invocados nos corpos dos métodos da classe com o número de métodos da classe, conforme sugerido por (CHIDAMBER e KEMERER, 1994).
Lack of Cohesion Methods	LCOM2	Calcula o conjunto de métodos em uma classe que não estão relacionados através da partilha de alguns dos atributos da classe. Em seguida, foi calculado o número de pares de métodos que compartilham. Por fim, a falta de coesão em métodos foi obtida subtraindo o número de pares de métodos que não compartilham um acesso de campo do numero de pares de métodos que compartilham.
Afferent Coupling	CA	Calcula a quantidade de classes que usam uma classe especifica. Essa medida, em CKJM Ext, foi usada para o cálculo de CBO.
Efferent Coupling	CE	Calcula a quantidade de classes que são usadas por uma classe especifica. Assim como a medida CE, também foi utilizada para o cálculo da CBO.
Lines of Codes	LOC	Calcula a soma do número de linhas de código de cada classe, desconsiderando linhas de comentários e linhas em branco.
Number of Class	nCL	Calcula contando o número de classes em cada *software* medido.

APÊNDICE D – SISTEMAS DE *SOFTWARE* MEDIDOS.

Tabela 58 Dados coletados (25 domínios, 3 versões de 107 sistemas de *software* e 237.370 classes)

Nome do *software*	**Versões**	**Data do lançamento**	**nCl**	**Domínio de aplicação**
aTunes (S1)	1.0	22/8/2006	355	*Audio e Video (D1)*
	2.0	20/4/2010	1204	
	3.0	15/12/2012	1789	
Subsonic (S2)	2.0	27/2/2005	34	
	3.0	22/3/2007	185	
	4.0	12/5/2010	297	
Impro-Visor (S3)	4.07	9/11/2009	1220	
	5.10	7/3/2012	1223	
	5.16	14/5/2012	1480	
StreamRipStar (S4)	0.3.0	26/7/2008	89	
	0.5.6	16/12/2009	125	
	0.6.4	25/10/2012	1006	
Music Box (S5)	1.01	8/2/2010	146	
	1.04	8/4/2010	203	
	1.07	8/9/2012	224	
Weka (S6)	3.0.6	15/2/2002	162	*Artificial Intelligence (D2)*
	3.3.6	17/4/2006	963	
	3.6.10	31/7/2013	2137	
Jenes - Genetic Algorithms for Java (S7)	1.2.0	9/2/2009	99	
	2.0.0	7/4/2012	226	
	2.1.0	8/1/2013	236	
Meka (S8)	0.85	2/9/2011	26	
	1.0	4/5/2012	34	
	1.3.0	8/3/2013	109	
Observation Manager (S9)	0.114	16/7/2006	94	*Astronomy (D3)*
	0.717	30/5/2008	150	
	1.221	10/3/2013	256	
Parallel Spectral Deconvolution (S10)	1.9	25/4/2009	183	
	1.11	27/8/2009	183	
	1.12	19/9/2013	184	
JDEread (S11)	1.0	12/3/2009	7	
	1.2	20/5/2010	6	
	1.4	26/7/2013	9	
Daylight Chart (S12)	1.0	16/4/2007	43	
	2.0	26/8/2007	113	
	3.3	8/9/2012	129	
Java TreeView (S13)	1.1.0	28/9/2006	305	*Bio-Informatics (D4)*
	1.1.4	8/9/2009	305	
	1.1.6r4	23/8/2013	342	
VarScan (S14)	2.2	27/4/2010	10	
	2.2.10	23/3/2012	14	
	2.3.6	29/7/2013	15	

Tabela 58, continua

TRMSim-WSN (S15)	0.2	15/5/2009	108	
	0.3.1	20/7/2009	119	
	0.5	25/3/2012	214	
DavMail (S16)	2.0.0	10/12/2008	76	*Business (D5)*
	3.0.0	12/2/2009	111	
	4.1.0	26/9/2012	284	
Freeplane (S17)	1.1.0	26/6/2010	193	
	1.1.3	6/12/2010	194	
	1.2.20	20/10/2012	197	
jBPM (S18)	3.0	24/6/2005	195	
	4.0	11/7/2009	689	
	5.0	4/2/2011	572	
Rapidminer (S19)	3.0	11/7/2005	1062	
	4.0	31/7/2007	1975	
	5.0	9/8/2010	4000	
PDF Split and Merge (S20)	1.0	29/6/2008	102	
	2.0	12/4/2010	114	
	2.2.1	31/1/2013	195	
GridSim (S21)	4.1	1/9/2007	175	*Clustering (D6)*
	5.0	24/9/2009	331	
	5.2	25/11/2010	340	
JGroups (S22)	2.2	15/9/2003	696	
	3.0	17/11/2011	801	
	3.2.6	17/1/2013	900	
JPPF (S23)	1.0	27/12/2007	337	
	2.1.1	4/5/2010	675	
	3.2.2	27/1/2013	1109	
Jacob (S24)	1.9	26/2/2005	18	
	1.13	6/10/2007	25	
	1.17	3/8/2013	29	
Azureus/Vulze (S25)	2.1.0	18/7/2003	162	*Communication (D7)*
	3.0.5.2	29/5/2008	5704	
	4.8.1.2	4/2/2013	8148	
Dbunit (S26)	2.4.0	27/11/2008	332	*DataBase (D8)*
	2.4.5	22/5/2009	351	
	2.4.9	30/9/2012	386	
Hibernate (S27)	2.0	10/6/2003	423	
	3.0	31/3/2005	957	
	4.0	15/12/2012	2722	
JabRef (S28)	1.0	30/11/2003	259	
	2.0	30/1/2006	1639	
	2.9.2	12/1/2013	5839	
Universal Password Manager (S29)	1.1	16/8/2011	51	
	1.6	16/8/2011	72	
	1.12	6/6/2013	92	
c3p0 JDBC Data Sources (S30)	0.8.5.1	2/3/2005	249	
	0.9.0	11/7/2005	310	
	0.9.2	9/2/2013	216	

Tabela 58, continua

Json-lib (S31)	1.0	12/2/2007	21	
	2.0	20/7/2007	51	
	2.4	14/12/2010	98	
PMD (S32)	4.1	17/11/2007	676	
	4.3	4/11/2011	733	
	5.0.2	3/2/2013	949	
VRaptor (S33)	3.0.1	22/10/2009	506	
	3.3.0	1/2/2011	507	
	3.4.1	10/4/2012	505	
Lightweight Java Game Library (S34)	2.0	13/10/2008	293	
	2.4	11/4/2010	303	
	2.8.5	4/11/2012	597	
JUnit (S35)	4.0	16/2/2006	92	
	4.10	19/8/2008	188	
	4.5	14/11/2012	252	
JHotDraw (S36)	5.2	19/2/2001	191	
	6.0	1/2/2004	398	
	7.0.6	6/9/2011	495	
JEdit (S37)	3.0	25/12/2000	26	
	4.0	12/4/2002	24	
	5.0	22/11/2012	48	
iReport (S38)	3.6.0	2/9/2009	2287	
	4.0.0	10/1/2011	2585	
	5.0.1	8/1/2013	2617	
Im4Java (S39)	1.0.0	22/3/2010	77	*Development (D9)*
	1.2.0	1/9/2011	98	
	1.4.0	27/12/2012	102	
Eclipse (core) (S40)	3.7.1	12/12/2011	384	
	4.2	28/6/2012	663	
	4.2.1	28/9/2012	1381	
DrJava (S41)	20090821-r5004	21/8/2009	4009	
	20100816-r5366	16/8/2010	5200	
	20120818-r5686	18/8/2012	5248	
Code Generation Library (S42)	2.1.3	13/10/2005	228	
	2.2	26/5/2008	226	
	2.2.3	27/7/2012	226	
Joda Time (S43)	1.0	22/2/2005	201	
	1.6	28/10/2008	222	
	2.3	16/8/2013	232	
jPDF Tweak (S44)	0.9	10/9/2007	87	
	1.0	27/12/2010	94	
	1.1	19/12/2011	110	
ProGuard (S45)	1.0	6/6/2002	127	
	3.5	24/1/2006	319	
	4.10	29/7/2013	576	
RText (S46)	1.0.0	4/12/2009	232	
	2.0.0	16/1/2012	218	
	2.0.7	3/5/2013	189	

Tabela 58, continua

Sahi (S47)	V2-20090521	21/5/2009	107	
	V3-20101103	3/11/2010	132	
	V44-20130429	29/4/2013	151	
Checkstyle (S48)	4.4	16/12/2007	309	
	5.2	25/9/2010	354	
	5.6	18/9/2012	364	
Cobertura (S49)	1.0	12/2/2005	46	
	1.5	5/8/2005	35	
	2.0	29/5/2013	147	
Demoiselle (S50)	1.1.0	11/1/2010	99	
	2.0.0	30/12/2010	91	
	2.3.0	31/10/2012	148	
Xmlsh (S51)	1.1.8	24/1/2012	336	
	1.2.0	14/6/2012	396	
	1.2.3	22/3/2013	372	*Development (D9)*
OmegaT (S52)	1.8.1	1/10/2009	334	
	2.3.0	28/4/2012	725	
	2.6.3	17/7/2013	880	
Gate (S53)	5.0	29/5/2009	2021	
	6.0	11/11/2010	2200	
	7.0	13/2/2012	2088	
Java Neural Network Neuroph (S54)	2.1	7/4/2009	71	
	2.4	8/6/2010	116	
	2.7	18/11/2012	157	
LaTeXDRAW (S55)	1.0.2	2/1/2006	60	
	1.9.3	24/6/2007	132	
	2.0.8	15/3/2010	250	
FreeMarker (S56)	2.03	12/6/2002	150	
	2.3.19	23/1/2007	583	
	2.3.9	1/3/2012	668	
iText (S57)	5.0.0	8/12/2009	469	
	5.1.0	6/5/2011	557	
	5.3.5	19/12/2012	706	*Dynamic Content (D10)*
Liferay (S58)	4.0	2/6/2006	830	
	5.0	9/4/2008	1560	
	6.0	4/6/2010	2090	
Wireless Universal Resource File (S59)	1.3.1	24/8/2011	185	
	1.4.1	15/8/2012	349	
	1.4.4.1	11/1/2013	359	
Jmol (S60)	11.0	8/5/2007	1257	
	12.0	4/10/2011	1584	
	13.0	25/1/2013	2056	
jMusic (S61)	1.2	26/11/2001	241	
	1.5	24/3/2004	297	*Education (D11)*
	1.6.4	5/2/2013	289	
Logisim (S62)	2.0	31/10/2005	908	
	2.4	30/7/2010	1185	
	2.7	7/3/2011	1347	

Tabela 58, continua

Cabra (S63)	0.4.2	20/3/2011	116	
	0.6.0	23/10/2011	142	
	0.7.0	29/10/2012	178	
Biogenesis (S64)	0.2.1	17/5/2006	29	*Education (D11)*
	0.5	12/11/2007	98	
	0.8	20/12/2010	97	
JStock (S65)	1.0	16/11/2008	383	
	1.0.5v	24/11/2010	784	
	1.0.6z	1/1/2013	866	
OpenBravoPos (S66)	2.0	25/2/2008	785	*Financial (D12)*
	2.2	29/8/2008	893	
	2.3	3/11/2010	999	
Buddi (S67)	1.0.0	23/6/2006	2533	
	2.6.3.0	8/9/2007	2613	
	3.4.1.7	17/4/2013	2294	
FreeCol (S68)	0.8	14/1/2009	927	
	0.9	1/1/2010	1081	
	0.10	11/6/2011	1200	
KoLmafia (S69)	13.0	5/3/2009	1036	
	14.0	19/3/2010	1194	
	15.0	20/1/2012	1368	
RoboCode (S70)	1.6.0.0	1/5/2008	374	
	1.7.0.2	15/2/2009	195	
	1.8.0.0	30/1/2013	196	
Hodoku (S71)	1.0	30/1/2009	305	
	2.0.1	14/4/2010	552	
	2.2.0	29/7/2012	591	*Games (D13)*
Vassal Engine (S72)	3.1.2	2/9/2012	2040	
	3.2.6	23/5/2013	2547	
	3.2.8	28/7/2013	2552	
TikiOne Steam Cleaner (S73)	1.0.1	28/1/2012	21	
	2.0.0	6/3/2013	60	
	2.2.0	20/6/2013	66	
PCGen (S74)	5.10.2	11/3/2007	1732	
	5.14.0	8/6/2008	1952	
	6.00.1	23/8/2013	3738	
FFGenie (S75)	2008.2	26/8/2008	291	
	2010.2	31/3/2010	372	
	2013.4	26/4/2013	384	
Art of Illusion (S76)	1.0	11/2/2002	319	
	2.0	8/3/2005	854	
	2.9.1	19/2/2012	719	
FreeMind (S77)	0.0.2	27/6/2000	55	*Graphics (D14)*
	0.6.0	1/2/2003	183	
	0.9.0	18/2/2011	877	
Jasper Reports (S78)	3.0	19/5/2008	1233	
	4.0	10/1/2011	1870	
	5.0	19/11/2012	2516	

Tabela 58, continua

JFreeChart (S79)	1.0.0	2/12/2005	481	*Graphics (D14)*
	1.0.14	4/1/2008	562	
	1.0.9	20/11/2011	620	
uclide (S80)	0.2.0	14/3/2007	126	
	0.5.3	4/8/2009	295	
	0.6.6	14/4/2012	354	
VietOCR (S81)	1.0	18/7/2009	138	
	2.0	3/10/2010	207	
	3.0	6/2/2011	187	
JChart2D (S82)	2.0.0	31/5/2006	177	
	3.0.0	1/6/2008	291	
	3.2.2	24/9/2011	355	
FractalToy (S83)	0.1.2	24/8/2010	155	
	0.2.0	5/1/2012	352	
	0.2.1	19/11/2012	357	
ForceFeedback Joystick Driver (S84)	0.5	7/11/2009	64	*Hardware (D15)*
	0.8	18/1/2010	56	
	0.8.3	19/11/2011	57	
Prime (S85)	1.0	18/3/2010	1861	
	3.0	19/7/2010	391	
	3.5	6/2/2012	2387	
Mobile Atlas Creator (S86)	1.9	18/8/2011	1549	*Mapping (D16)*
	1.9.7	16/5/2012	1598	
	1.9.14	29/7/2013	1635	
GeoAPI (S87)	1.0.0	18/5/2004	249	
	2.0.0	8/6/2005	479	
	3.0.0	3/6/2011	259	
IP Monitor (S88)	0.01	28/8/2007	69	*Networking (D17)*
	0.03	20/3/2008	115	
	0.04	27/3/2010	169	
Java Network Browser (S89)	1.0	23/2/2003	74	
	1.28	10/5/2003	627	
	1.56	30/5/2012	2007	
SoapUI (S90)	2.0	12/12/2007	1808	
	3.0	9/7/2009	2590	
	4.0	14/6/2011	3225	
Yet Another Java Service Wrapper (S91)	9.4	19/9/2009	501	
	10.4	10/10/2010	820	
	11.4	11/1/2013	489	
jNetPcap (S92)	1.0	26/12/2007	19	
	1.2	19/3/2009	292	
	1.4	21/8/2013	383	
Kalender (S93)	1.1	9/6/2012	50	*Scheduling (D18)*
	1.6	8/9/2012	58	
	2.0	31/5/2013	74	
Jasypt (S94)	1.4.1	16/1/2008	105	*Security (D19)*
	1.7	29/10/2010	131	
	1.9.0	18/12/2011	100	

Tabela 58, conclusão

JSch (S95)	0.1.0	11/2/2003	80	
	0.1.25	16/2/2006	93	
	0.1.49	11/10/2012	129	
JXplorer (S96)	3.0	3/11/2003	364	*Security (D19)*
	3.2	3/3/2007	399	
	3.3.02	10/11/2012	412	
JFileCrypt (S97)	0.1.2	3/1/2006	17	*Storage (D20)*
	0.2.0	24/9/2006	33	
	0.3.0	13/6/2010	163	
Areca Backup (S98)	5.0	26/5/2007	345	
	6.1	20/12/2008	517	
	7.3.7	31/8/2013	792	
DocFetcher (S99)	0.8.0	22/8/2007	135	
	1.0	12/5/2009	339	
	1.0.3	18/3/2010	334	
JeeObserver (S100)	2.0.0	15/12/2009	95	*Test and Measurement (D21)*
	2.1.2	2/6/2010	101	
	3.1.2	16/3/2012	181	
Zephyr (S101)	2.3.2	30/10/2007	382	
	2.7	23/9/2009	389	
	2.8	11/9/2012	404	
DataCrow (S102)	2.4	16/7/2006	516	*Utilities (D22)*
	3.2	24/3/2008	658	
	3.9.20	2/2/2013	795	
FileBot (S103)	2.0	15/9/2011	7594	
	2.65	24/6/2012	8353	
	3.3	18/1/2013	11502	
Med's movie (S104)	1.70	24/2/2005	73	
	2.3	22/4/2006	592	
	2.9	31/1/2010	188	
Google2SRT (S105)	0.1	31/8/2008	15	*Video (D23)*
	0.3	31/8/2008	21	
	0.6	11/8/2013	40	
Java YouTube vídeo Downloader (S106)	V20101219	19/12/2010	10	*WWW/HTTP (D24)*
	V20110829	29/8/2011	14	
	V20130506	6/5/2013	15	
ZK Simply Ajax and Mobile (S107)	5.0.3	30/6/2010	485	*Protocols/AJAX (D25)*
	6.0.0	14/2/2012	632	
	7.0.0	13/8/2013	656	

APÊNDICE E – *POST-HOC* DOS SISTEMAS DE *SOFTWARE*.

Tabela 59 Teste *post-hoc* de comparações múltiplas de *Kruskall-Wallis* dos sistemas de *software* da versão 1.

Software	WMC	DIT	NOC	CBO	RFC	LCOM	CA	CE	LOC
1	18	41	0	32	10	31	10	21	18
2	5	3	0	2	3	0	2	1	5
3	83	59	3	89	90	80	34	73	83
4	14	50	0	66	7	38	4	34	14
5	16	41	0	79	8	7	15	44	16
6	85	46	0	55	77	56	13	36	85
7	8	25	0	18	7	7	4	7	8
8	45	29	0	23	55	2	10	57	45
9	18	7	0	13	8	10	4	5	18
10	40	17	0	18	27	45	7	13	40
11	6	0	0	0	0	0	0	0	6
12	3	3	0	4	2	1	3	0	3
13	17	55	0	52	13	29	9	31	17
14	53	0	0	25	7	0	0	1	53
15	43	15	0	54	46	30	4	44	43
16	10	13	0	9	6	3	4	8	10
17	11	39	0	16	8	12	11	15	11
18	11	25	0	19	11	14	12	14	11
19	30	39	0	30	22	25	22	33	30
20	15	51	0	14	22	3	5	7	15
21	50	34	0	23	27	22	7	11	50
22	39	31	0	34	34	18	22	24	39
23	18	25	0	27	12	13	20	25	18
24	1	1	0	1	3	18	0	0	1
25	9	17	0	22	8	36	6	30	9
26	22	64	0	74	26	14	13	63	22
27	20	66	0	48	21	58	22	27	20
28	13	72	0	29	8	50	7	18	13
29	11	4	0	37	5	13	3	22	11
30	16	21	0	44	9	11	8	46	16
31	1	0	0	5	1	3	0	0	1
32	25	61	0	41	16	27	14	27	25
33	30	32	0	28	26	24	26	22	30
34	19	24	0	57	20	41	81	43	19
35	16	19	0	12	10	2	4	5	16
36	14	46	0	36	9	20	12	21	14
37	1	2	0	16	2	0	0	1	1
38	43	41	3	50	32	24	43	30	43
39	4	73	0	11	6	2	7	7	4
40	23	29	0	77	26	57	73	31	23
41	40	42	2	39	35	30	22	31	40
42	31	26	0	25	26	14	11	18	31
43	54	60	0	70	72	63	30	61	54
44	4	14	0	17	4	11	4	5	4
45	12	15	0	77	7	16	49	26	12
46	13	25	0	20	11	10	7	18	13
47	21	9	0	13	34	24	6	8	21
48	24	72	0	30	18	35	59	18	24
49	5	1	0	4	7	14	3	1	5
50	32	32	0	27	19	3	10	38	32
51	22	59	0	36	26	15	50	38	22
52	35	48	0	66	40	31	12	57	35
53	27	49	0	38	23	31	16	29	27
54	6	54	0	26	6	2	8	3	6

Tabela 59, conclusão

55	74	8	0	26	72	6	20	4	74
56	11	25	0	30	8	8	6	20	11
57	54	36	0	53	28	25	30	30	54
58	68	39	2	67	17	88	88	59	68
59	41	27	0	24	27	11	7	14	41
60	63	42	0	37	35	27	18	29	63
61	31	24	0	46	10	13	21	23	31
62	37	34	0	36	29	28	22	28	37
63	8	49	0	19	7	7	5	7	8
64	1	33	0	10	1	1	0	1	1
65	23	49	0	40	25	31	11	27	23
66	39	41	0	34	40	26	15	30	39
67	30	38	3	37	23	32	20	35	30
68	26	35	0	44	18	19	18	47	26
69	30	58	0	48	28	25	27	57	30
70	17	33	0	26	12	15	50	34	17
71	68	51	0	85	79	66	23	65	68
72	37	38	0	39	33	27	15	30	37
73	3	11	0	21	0	1	0	1	3
74	40	39	0	48	33	37	24	34	40
75	33	25	0	30	22	13	9	18	33
76	68	34	0	49	33	13	31	40	68
77	4	47	0	6	3	2	2	8	4
78	22	62	3	68	19	45	18	52	22
79	41	46	0	48	34	46	13	24	41
80	21	74	0	19	8	36	84	30	21
81	30	56	0	86	50	50	16	66	30
82	16	21	0	37	14	20	8	35	16
83	8	29	0	26	7	7	6	14	8
84	18	21	0	8	20	2	3	4	18
85	28	37	0	46	25	30	21	42	28
86	27	36	0	45	20	27	44	29	27
87	102	22	0	47	98	11	14	59	102
88	10	14	0	16	6	8	5	16	10
89	6	14	0	9	3	5	3	11	6
90	25	37	0	51	21	23	15	57	25
91	69	31	0	50	62	21	37	37	69
92	1	0	0	36	1	9	1	44	1
93	9	75	0	8	6	2	4	11	9
94	13	14	0	17	6	5	26	12	13
95	8	5	0	12	6	9	4	8	8
96	22	43	0	36	14	24	11	25	22
97	0	0	0	8	0	0	0	0	0
98	26	30	0	49	29	22	27	29	26
99	10	25	0	54	8	4	10	59	10
100	11	13	0	25	6	6	5	60	11
101	22	30	0	37	24	15	10	23	22
102	23	29	0	30	24	21	17	21	23
103	28	60	0	39	29	56	36	36	28
104	6	53	0	51	3	10	4	34	6
105	0	0	0	0	0	0	0	0	0
106	0	2	0	4	0	0	0	0	0
107	36	35	0	39	26	32	22	25	36

Tabela 60 Teste *post-hoc* de comparações múltiplas de *Kruskall-Wallis* dos sistemas de *software* da versão 2.

Software	**WMC**	**DIT**	**NOC**	**CBO**	**RFC**	**LCOM**	**CA**	**CE**	**LOC**
1	33	42	0	42	28	27	24	39	40
2	51	23	0	28	36	55	18	14	39
3	73	68	4	91	90	84	46	83	85
4	33	59	0	70	7	46	9	40	14
5	46	59	0	88	22	23	26	59	33
6	32	56	0	46	32	28	22	31	47
7	15	33	0	32	8	18	7	20	29
8	4	48	0	25	52	1	12	49	46
9	8	21	0	19	34	19	6	12	46
10	59	23	0	22	41	53	7	14	50
11	0	0	0	1	0	0	0	0	6
12	21	20	0	16	7	5	10	16	9
13	25	66	0	58	11	34	13	40	23
14	2	0	0	47	41	0	0	11	86
15	23	16	0	58	43	24	4	46	42
16	32	21	0	43	17	9	4	17	23
17	11	54	0	26	6	17	14	21	14
18	31	72	0	49	34	24	28	36	48
19	37	50	0	48	35	31	37	52	48
20	8	56	0	19	27	5	4	18	21
21	70	31	0	32	36	19	14	26	58
22	48	40	0	38	45	28	19	31	49
23	29	37	0	49	25	26	36	29	33
24	0	20	0	1	1	9	0	0	1
25	42	51	4	64	45	40	33	67	49
26	52	69	1	76	35	20	17	68	29
27	59	65	4	82	29	65	48	61	35
28	33	57	0	49	30	35	24	34	38
29	22	25	0	54	8	26	2	46	24
30	16	31	0	47	15	22	19	46	27
31	5	18	0	7	3	17	2	5	2
32	28	70	0	52	23	45	18	33	31
33	40	34	0	37	55	26	40	27	67
34	15	32	0	67	30	51	91	49	29
35	20	33	0	23	40	11	8	25	48
36	49	56	1	48	25	52	18	34	21
37	2	5	0	11	1	1	0	0	1
38	38	42	4	55	44	33	51	36	52
39	6	74	0	12	5	4	11	16	7
40	59	38	0	84	25	50	69	54	31
41	38	50	4	49	44	34	33	41	46
42	26	22	0	29	39	21	12	19	40
43	97	75	0	80	87	71	44	81	71
44	11	20	0	30	5	17	2	8	7
45	71	31	0	103	28	57	74	82	32
46	24	29	0	25	12	13	7	18	13
47	44	19	0	17	48	38	4	16	40
48	43	80	0	35	23	48	69	23	37
49	8	1	0	5	15	12	0	0	12
50	5	8	0	47	22	4	42	61	38
51	26	64	0	40	28	21	61	40	28
52	37	56	0	68	50	31	20	64	50
53	48	58	1	47	32	52	26	35	38
54	9	55	0	22	14	9	5	9	28
55	58	27	0	28	83	12	50	11	74
56	29	40	0	50	38	26	19	55	41

Tabela 60, conclusão

57	53	36	0	50	27	32	36	32	57
58	89	42	10	55	40	99	90	43	63
59	25	71	0	32	43	19	20	24	51
60	54	48	4	48	46	30	26	34	77
61	16	30	0	56	10	21	20	36	40
62	60	41	0	40	46	39	23	33	52
63	25	57	0	25	6	11	5	14	12
64	28	15	0	30	9	75	3	9	8
65	46	64	1	57	50	50	20	49	48
66	31	51	0	44	50	37	21	31	48
67	46	42	7	48	31	48	27	46	42
68	32	41	0	53	28	31	26	57	37
69	36	65	0	59	46	34	43	66	50
70	16	19	0	31	55	24	11	31	79
71	79	69	0	91	91	80	31	79	85
72	40	43	0	48	43	36	23	37	46
73	22	47	0	42	5	39	1	16	9
74	47	55	1	53	45	44	27	37	48
75	28	45	0	38	34	24	11	24	40
76	36	54	0	76	38	29	47	68	62
77	27	64	0	22	7	21	7	17	23
78	37	63	1	78	32	51	25	64	38
79	81	58	1	53	48	56	18	32	61
80	24	92	0	62	28	62	65	72	47
81	65	65	0	90	48	62	19	73	41
82	35	31	0	32	14	24	14	28	29
83	20	31	0	37	26	14	15	26	24
84	5	21	0	11	3	11	2	10	5
85	27	38	0	45	34	15	17	27	47
86	35	41	1	53	29	33	51	34	39
87	58	34	0	68	103	25	26	73	104
88	7	25	0	30	7	9	11	24	24
89	59	39	0	37	40	50	25	28	49
90	38	44	0	62	33	33	25	69	43
91	40	41	0	44	38	31	24	29	45
92	48	29	0	34	13	53	11	35	15
93	5	85	0	11	6	4	7	19	19
94	10	17	0	25	7	14	51	21	14
95	26	6	0	13	4	16	3	17	7
96	26	57	0	48	22	34	13	36	31
97	0	18	0	2	4	1	2	5	4
98	47	37	0	56	40	24	40	35	38
99	35	72	0	45	17	36	12	81	28
100	11	7	0	44	10	12	7	73	10
101	25	36	0	44	31	15	13	27	31
102	49	39	0	63	64	42	20	55	52
103	55	65	1	49	39	58	44	39	45
104	55	75	0	52	26	68	18	31	31
105	25	30	0	37	36	6	0	20	31
106	0	2	0	5	0	0	0	0	0
107	29	37	0	49	41	39	33	32	47

Tabela 61 Teste *post-hoc* de comparações múltiplas de Kruskall-Wallis dos sistemas de *software* da versão 3.

Software	WMC	DIT	NOC	CBO	RFC	LCOM	CA	CE	LOC
1	41	40	0	48	36	54	48	44	48
2	57	25	0	29	45	60	20	28	46
3	65	72	2	93	85	84	52	79	80
4	69	46	0	43	76	54	23	40	83
5	83	87	0	83	38	89	42	54	41
6	42	43	0	51	57	43	28	61	61
7	13	33	0	38	12	18	9	25	36
8	17	28	0	18	31	10	13	29	34
9	15	32	0	32	46	30	9	24	49
10	59	24	0	26	35	57	7	18	54
11	1	3	0	17	0	0	4	5	37
12	11	16	0	20	7	6	16	15	16
13	31	71	0	63	21	40	12	44	25
14	2	0	0	53	59	0	0	13	97
15	25	23	0	62	41	41	7	50	37
16	46	27	0	30	36	22	16	25	24
17	10	64	0	26	10	17	15	19	14
18	24	63	0	71	22	33	18	51	27
19	42	62	1	55	43	35	35	63	51
20	18	70	0	29	49	13	9	18	34
21	79	28	0	32	48	19	14	26	59
22	49	38	0	40	56	30	20	36	54
23	36	38	0	45	37	34	21	44	47
24	2	14	0	3	2	8	0	4	1
25	58	49	10	71	49	49	31	83	50
26	53	72	1	79	44	16	17	75	37
27	48	49	2	72	43	57	68	47	50
28	42	44	2	49	42	38	31	45	50
29	24	27	0	48	19	30	3	39	32
30	18	23	0	31	18	10	15	25	15
31	5	20	0	13	12	19	3	10	21
32	28	83	0	67	31	56	25	56	36
33	49	39	0	39	60	31	44	31	80
34	26	37	0	63	40	60	90	53	46
35	33	33	0	33	51	20	11	31	65
36	37	34	0	38	50	26	21	35	43
37	3	19	0	19	2	1	2	13	6
38	41	42	5	57	46	36	59	42	59
39	6	76	0	13	6	5	11	14	9
40	47	50	1	78	37	40	53	76	38
41	36	47	5	50	44	35	35	49	50
42	30	23	0	27	34	21	12	24	41
43	103	69	0	87	90	79	41	88	73
44	12	25	0	28	6	17	5	15	10
45	63	82	0	105	34	31	62	98	43
46	25	22	0	25	17	9	7	27	14
47	45	22	0	19	53	45	6	18	41
48	47	81	0	33	33	48	74	26	42
49	37	19	0	36	12	14	5	58	34
50	16	19	0	48	51	11	29	58	63
51	27	64	0	41	37	19	60	46	36
52	43	67	0	59	51	33	22	58	50
53	52	70	0	49	36	49	29	41	39
54	9	55	0	25	10	10	12	18	17
55	65	24	0	61	75	46	92	27	82
56	35	49	0	55	35	27	17	56	46

Tabela 61, conclusão

57	55	34	0	55	36	35	37	38	60
58	102	40	15	50	67	96	100	42	50
59	18	70	0	36	41	22	21	29	55
60	48	40	4	47	42	36	29	39	69
61	21	32	0	69	22	22	22	44	40
62	72	40	0	45	58	40	25	38	61
63	33	72	0	28	9	20	9	19	19
64	24	13	0	26	7	75	3	10	8
65	55	73	0	64	55	56	21	55	48
66	36	68	0	46	51	36	21	33	50
67	49	42	3	49	41	44	29	49	50
68	37	38	0	67	38	34	33	76	45
69	36	70	0	68	57	38	47	82	54
70	13	22	0	29	55	26	13	32	81
71	83	80	0	95	93	89	40	79	87
72	47	42	0	48	44	41	22	42	51
73	26	54	0	47	7	56	2	21	16
74	45	50	1	47	42	41	22	49	50
75	33	49	0	35	32	23	12	28	42
76	44	55	0	88	54	35	37	97	70
77	32	63	0	40	31	31	20	42	38
78	41	62	3	88	41	54	29	73	43
79	88	60	1	65	57	60	16	39	62
80	18	94	0	70	36	62	76	86	57
81	61	77	0	93	51	64	23	77	41
82	41	31	0	36	20	32	14	25	24
83	19	28	0	35	27	17	15	25	25
84	5	19	0	9	5	8	2	12	5
85	40	42	2	50	40	43	22	46	47
86	30	40	2	61	34	32	53	41	48
87	44	29	0	75	102	19	60	75	105
88	13	28	0	23	21	10	15	27	41
89	40	58	2	45	37	38	24	39	50
90	40	40	0	71	43	38	23	83	46
91	51	34	0	39	48	33	21	28	47
92	46	26	0	63	23	61	54	61	24
93	6	92	0	21	10	3	9	31	12
94	6	20	0	29	16	15	12	33	23
95	18	18	0	19	9	14	5	20	10
96	26	67	0	44	30	41	15	34	38
97	47	23	0	28	25	46	7	41	51
98	29	37	0	67	37	24	46	54	43
99	45	54	0	51	22	41	11	88	31
100	32	21	1	50	21	11	8	77	13
101	24	31	0	46	36	19	12	38	38
102	66	37	0	84	85	45	24	81	63
103	49	67	3	52	44	58	45	44	49
104	31	87	0	64	16	43	5	50	15
105	40	56	0	69	57	15	0	55	43
106	0	15	0	25	0	0	0	12	1
107	27	32	0	52	39	45	33	40	48

APÊNDICE F – FREQUÊNCIA DAS MEDIDAS SELECIONADAS.

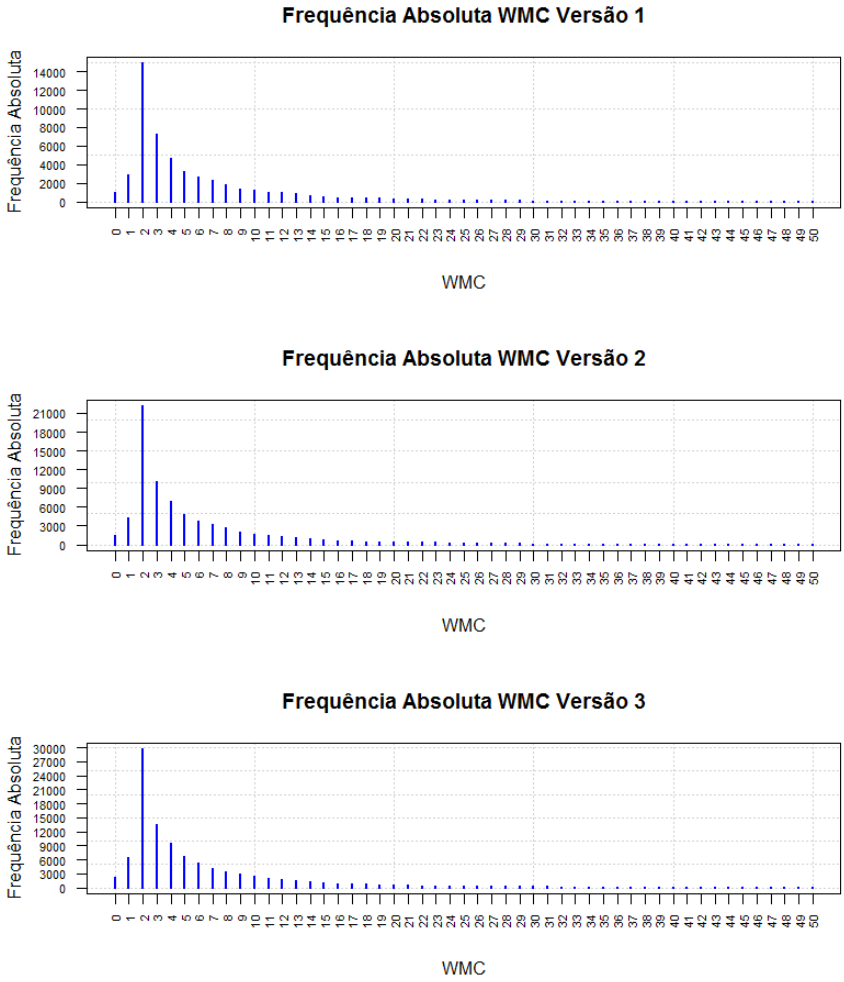

Figura 15 Gráficos de frequência absoluta da medida WMC, considerando as três versões de sistemas de *software* estudados.

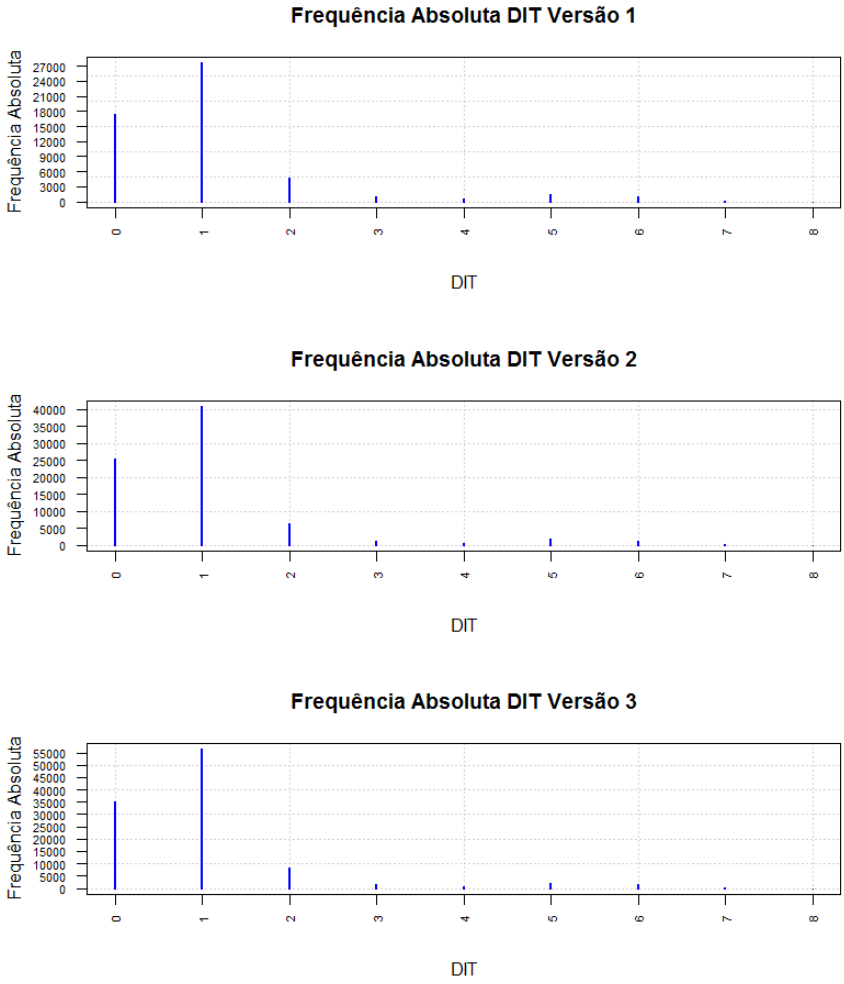

Figura 16 Gráficos de frequência absoluta da medida DIT, considerando as três versões de sistemas de *software* estudados.

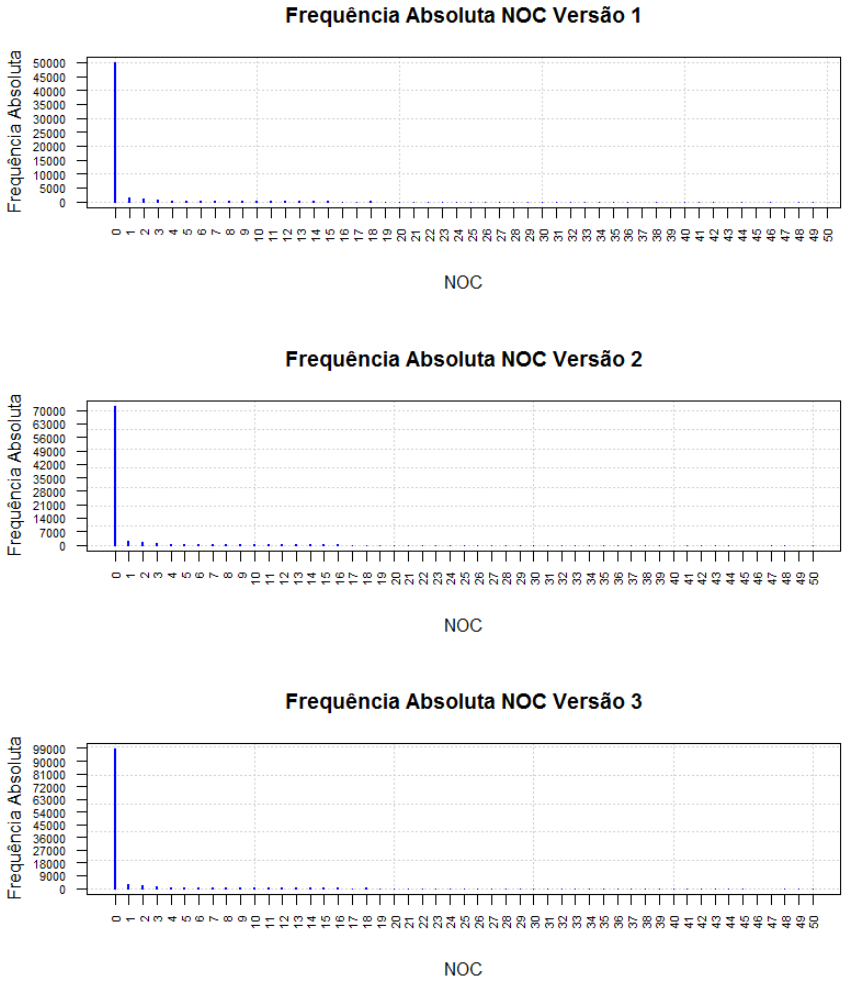

Figura 17 Gráficos de frequência absoluta da medida NOC, considerando as três versões de sistemas de *software* estudados.

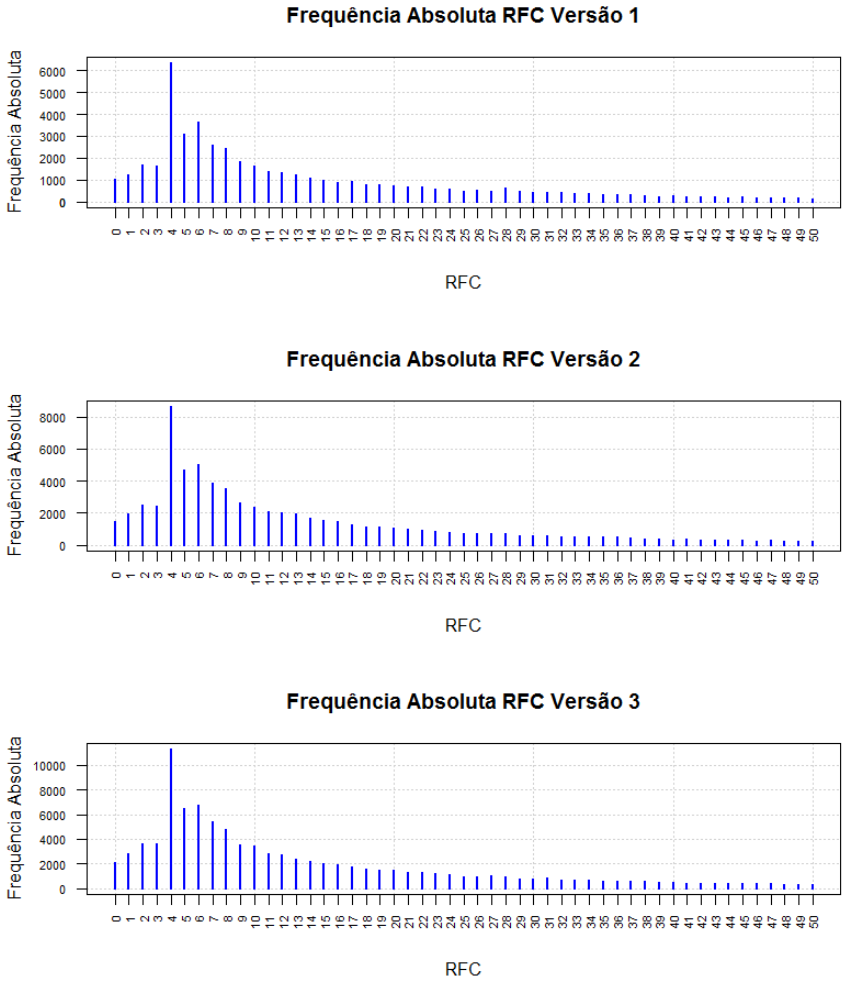

Figura 18 Gráficos de frequência absoluta da medida RFC, considerando as três versões de sistemas de *software* estudados.

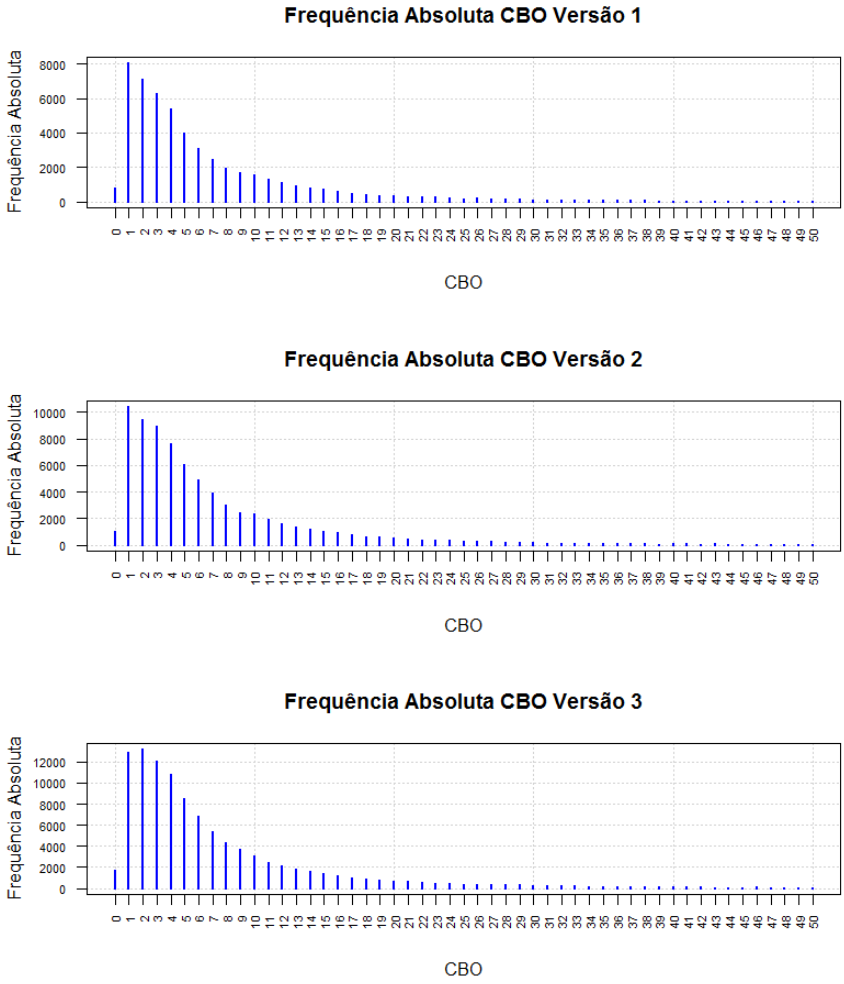

Figura 19 Gráficos de frequência absoluta da medida CBO, considerando as três versões de sistemas de *software* estudados.

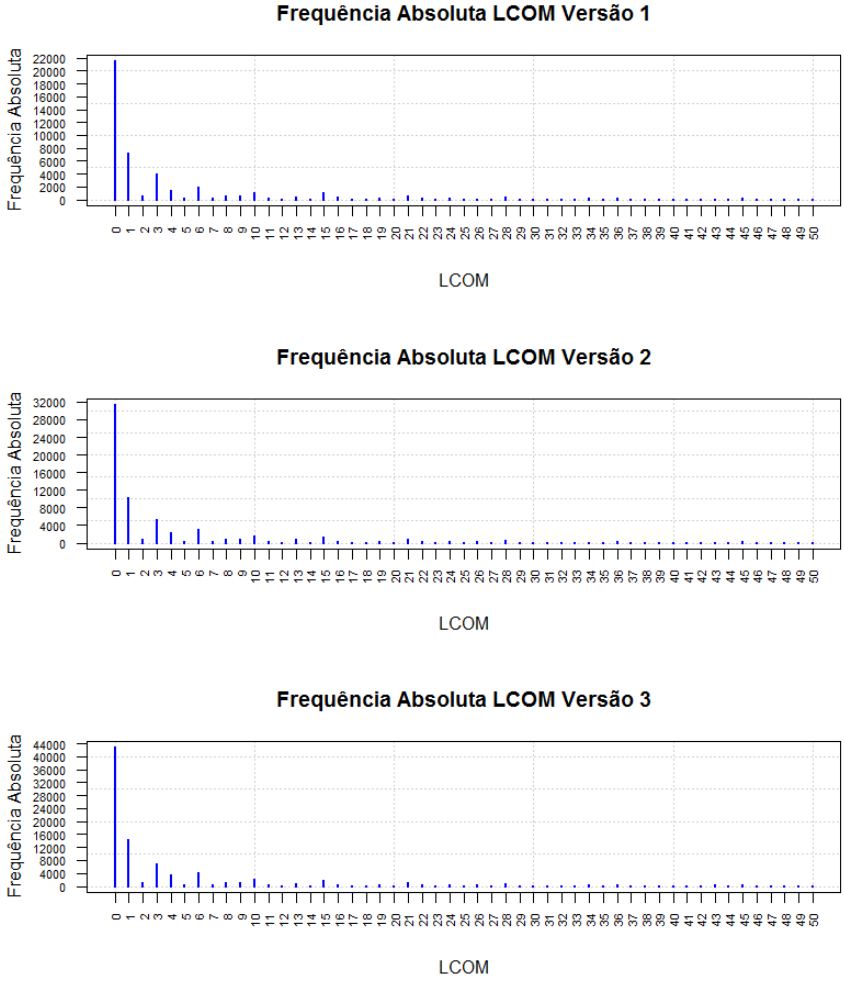

Figura 20 Gráficos de frequência absoluta da medida LCOM, considerando as três versões de sistemas de *software* estudados.

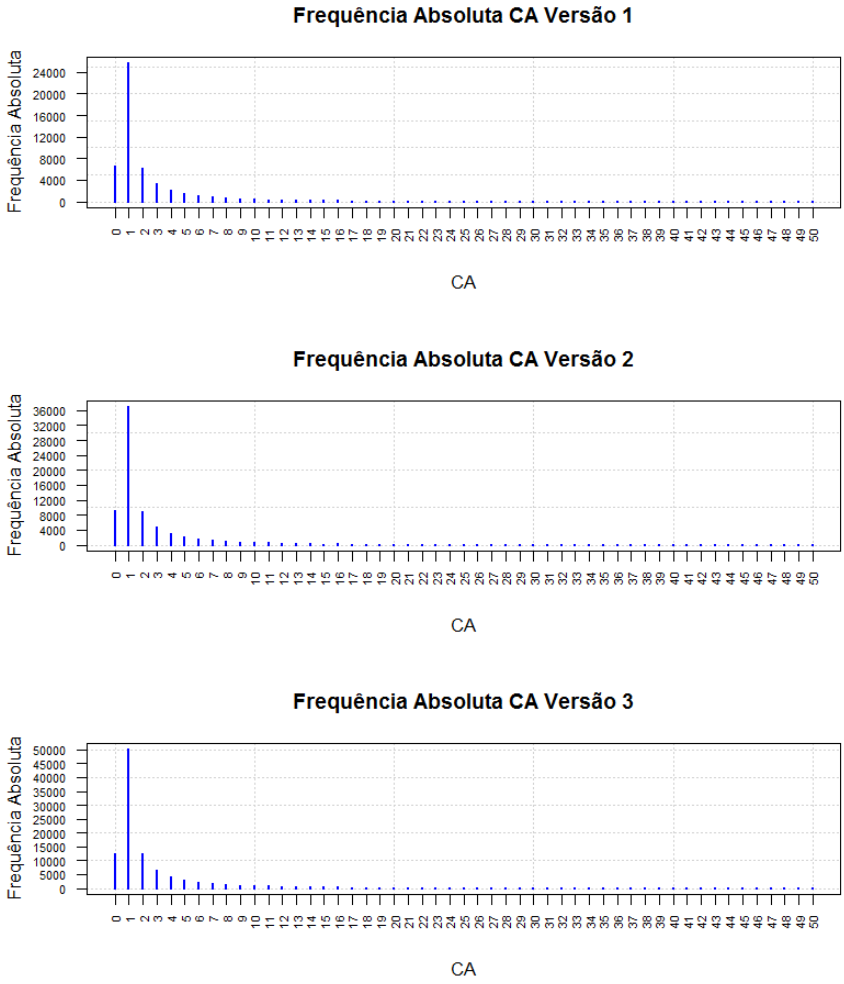

Figura 21 Gráficos de frequência absoluta da medida CA, considerando as três versões de sistemas de *software* estudados.

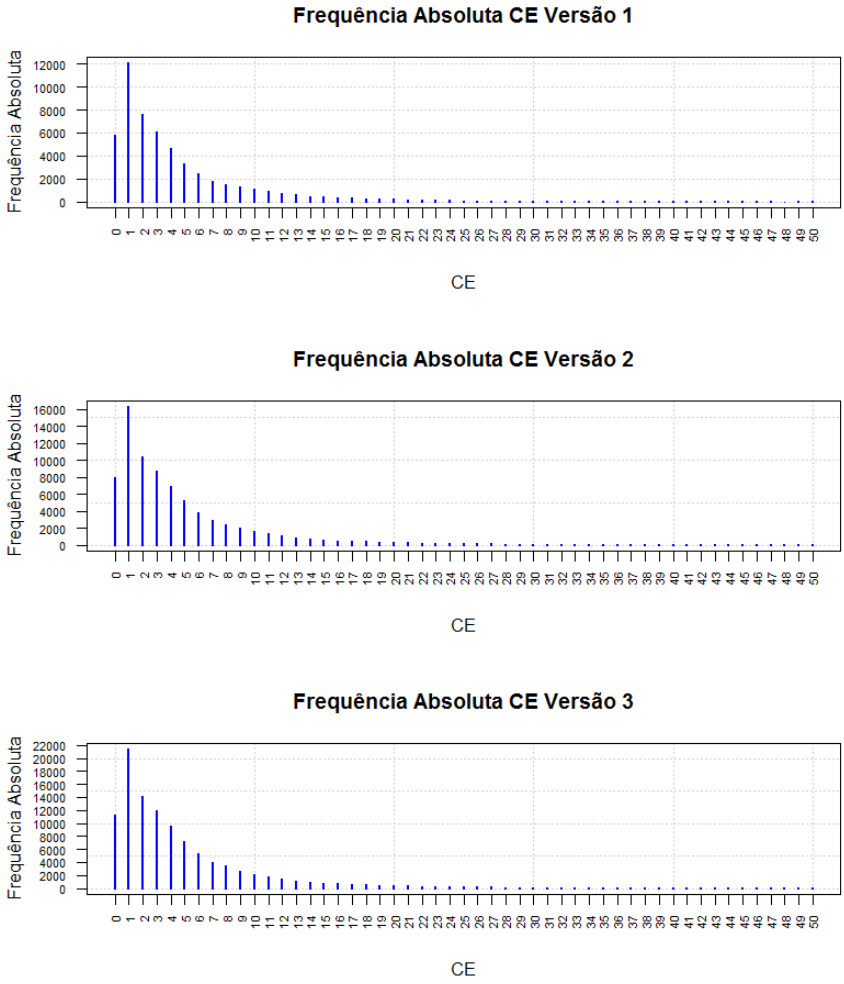

Figura 22 Gráficos de frequência absoluta da medida CE, considerando as três versões de sistemas de *software* estudados.