Reference Values for Object-Oriented Software Metrics

Kecia A. M. Ferreira, Mariza A. S. Bigonha, Roberto S. Bigonha
Heitor C. Almeida, Luiz F. O. Mendes
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais(UFMG)
Belo Horizonte, Brazil
Email: {kecia,mariza,bigonha,heitorca,lfmendes}@dcc.ufmg.br

***Resumo*—Embora uma grande quantidade de software OO tenha sido produzida, pouco se sabe sobre a real estrutura deste tipo de software. Há um grande número de métricas de software OO propostas, mas elas ainda não são empregadas de forma efetiva na indústria. Um motivo para isso é que há poucos dados publicados sobre este assunto e os valores típicos das medidas dessas métricas não são conhecidos. Este artigo apresenta os resultados de um estudo realizado com um grande conjunto de softwares abertos desenvolvidos em Java. O objetivo deste estudo foi identificar características deste tipo de software em termos de um conjunto de métricas de software OO, tais como grau de conectividade, grau de coesão de classes e profundidade das classes na árvore de herança. Os resultados do estudo sustentam uma compreensão importante sobre a estrutura de software aberto OO e evidenciam valores que podem ser tomados como referência para as medidas das métricas.**

***Abstract*—Although a large quantity of OO software has been produced, little is known about the actual structure of this type of software. There is a large number of proposed metrics for OO software, but they are still not employed effectively in industry. A reason for this is that there are few data published about this topic, and typical values of the metrics are not known. This paper presents the results of a study carried out on a large collection of open-source software developed in Java. The objective of this study was to identify characteristics of this type of software in terms of a set of metrics for OO software, such as connectivity, class cohesion and depth of a class in its inheritance tree. The results of the study provide important insights on the structure of open-source OO software and exhibit values that can be taken as baselines for the values of measures of the metrics.**

***Keywords*-software metrics; object-oriented software; probability distribution;**

I. Introdução

Embora a importância de métricas na produção de software seja notória, uma vez que, por meio delas, é possível medir, avaliar, controlar e melhorar os produtos e processos, este recurso não tem sido amplamente utilizado na indústria. Tempero [13] avalia que um dos motivos para esse fato é que para a maior parte das métricas de software ainda não se conhece o modelo de entidade da população (*entity population model*), o qual refere-se aos valores típicos de medidas que uma métrica assume em determinado conjunto de entidades. Há uma carência de estudos empíricos sobre o assunto que gerem resultados que possam ser aplicados a software de forma geral [13].

A caracterização de softwares OO é importante, porque pouco se conhece sobre as características de softwares reais produzidos neste paradigma. Não se conhece, por exemplo, informações simples como quantos métodos uma classe típica possui, ou quantas classes um pacote típico possui. Muitas métricas de software OO têm sido propostas [1], [3], [4], [7], [15], porém pouco se sabe sobre as suas medidas em softwares reais. Além disso, ainda está em aberto a definição dos valores desejáveis para essas métricas, o que restringe o uso efetivo de métricas na produção de software.

Um problema recorrente na área de medição de software é a obtenção de dados para a realização dos experimentos, pois a coleta de medidas envolve muitas vezes a análise de código fonte ou compilado dos softwares. Uma solução para isso é analisar dados de software livre. Há hoje disponível uma grande quantidade de softwares desta natureza. Sourceforge (www.sourceforge.net), por exemplo, conta com mais de 176.000 softwares livres registrados.

Este trabalho visa realizar um estudo de caracterização de softwares OO. O propósito deste estudo é obter resultados que possam evidenciar as características deste tipo de software sob os aspectos de grau de conectividade do software, grau de coesão das classes, profundidade na árvore de herança das classes e número de conexões aferentes das classes, que corresponde ao total de classes que usam serviços de uma determinada classe, além do número de atributos e de métodos públicos. Busca-se identificar valores a serem considerados como referência para um conjunto de métricas de software OO que avaliam esses aspectos. Para isso, investigou-se se as métricas descritas na Seção III-A podem ser modeladas por alguma distribuição de probabilidades. É importante identificar o modelo probabilístico adequado a um fenômeno porque isso viabiliza o uso de probabilidades em fenômenos reais. Por exemplo, para determinar o número adequado de atendentes em uma central telefônica, é importante conhecer a distribuição de probabilidades que modela o número de clientes que ligam para a central em um determinado intervalo de tempo [12]. No caso

978-0-7695-3844-0/09 $25.00 © 2009 IEEE
DOI 10.1109/SBES.2009.27

de software, a identificação do modelo probabilístico das característica avaliadas neste trabalho auxilia, por exemplo, na determinação de emprego de recursos na manutenção do software. Além disso, a partir da identificação desse modelo, é possível identificar valores típicos da característica avaliada ou, pelo menos, concluir que não há um valor típico.

II. Trabalhos Relacionados

Uma crescente atenção tem sido dada por pesquisadores à análise da forma como os módulos de um software conectam-se entre si e tem-se chegado à conclusão que softwares parecem ser regidos pelas chamadas *power laws* [2], [6], [9], [10], [16]. Uma *power law* é uma função de distribuição de probabilidades na qual a probabilidade de uma variável randômica *X* assumir um valor *x* é proporcional a uma potência negativa de x. Essa relação é mostrada na Equação 1.

$$P(X = x) \propto cx^{-k} \tag{1}$$

Uma distribuição que segue uma *power law* é dita ser de cauda pesada (*heavy-tail distribution*), sendo caracterizada por existir uma quantidade não desprezível de ocorrências em que a variável randômica assume valores muito altos, mas na maior parte das ocorrências assume valores baixos. Outra característica desta distribuição é ter escala livre (*scale-free*), o que significa que a média não é um valor informativo, ou seja, não há um valor que possa ser considerado típico para a característica avaliada. Uma grande variedade de fenômenos são modelados por *power laws*, por exemplo: frequência de uso de palavras, citações de artigos científicos, chamadas telefônicas, graus de entrada e saída dos nós na WWW e na Internet [8].

O relacionamento entre as classes ou entre os objetos de um sistema OO pode ser modelado como um grafo dirigido. Alguns pesquisadores têm utilizado esta modelagem e identificaram que esses grafos seguem *power laws* e, consequentemente, têm características de redes de livre escala, o que significa que não há um valor típico para determinada característica da rede. Potanin et al. [9] identificaram que a geometria do grafo que representa o relacionamento entre os objetos de um software OO, em tempo de execução, tem escala livre. Um grafo com geometria em escala livre se diferencia de um grafo no qual as arestas são randomicamente distribuídas. Em um grafo randômico, os nodos têm aproximadamente o mesmo grau, ou seja, a média dos graus dos nodos é um valor significativo. A Web é um exemplo de um grafo em escala livre porque não há um valor médio para a conectividade de seus nodos. Foram analisados 60 grafos de objetos de 35 programas. De acordo com Potanin et al. [9], as conclusões do trabalho são úteis para auxiliar a melhora de desempenho de coletores de lixo, a depuração de programas e o desempenho de programas.

Wheelson e Counsell [16] identificaram *power laws* nos relacionamentos entre as classes de software OO desenvolvido em Java. A análise realizada por eles teve como amostra três softwares bem conhecidos: JDK (*Java Development Kit*), Apache Ant e Tomcat, totalizando 6.870 classes. O objetivo do trabalho é verificar *power laws* nos diferentes tipos de conexões que podem ocorrer em um software OO: herança, implementação de interface, agregação (classes que têm outras classes como tipos de atributos e classes que são tipos de atributos de outras classes), uma classe fazer parte da lista de parâmetros de métodos de outra classe, uma classe ser tipo de retorno de métodos de outra classe. Além disso, verificou-se que as seguintes características de classes também seguem *power laws*: número de campos, métodos e construtores em cada classe.

O trabalho de Louridas et al. [6] analisa as distribuições estatísticas que os graus de entrada e de saída dos módulos de um sistema seguem. A amostra de dados analisada corresponde a softwares desenvolvidos em C, Perl, Java e Ruby. Foram analisados 11 softwares, dentre eles: J2SE SDK, Eclipse, OpenOffice e Ruby 1.8.2. O trabalho concluiu que, independente do paradigma de programação, os graus de entrada e saída são regidos por *power law*, ou seja, há uma grande número de módulos que possuem baixo grau de entrada e saída, e um número pequeno, mas não insignificante, que possui valores muito altos desses graus.

Os achados desses trabalhos têm grande importância, porém pode ser precipitado afirmar categoricamente que os seus resultados são conclusivos, pois as amostras investigadas podem não representar significativamente o universo de softwares existentes. Por exemplo, a maior parte dos trabalhos baseia-se em softwares abertos, conhecidos e diretamente relacionados à area de desenvolvimento de software e pesquisa. Pode ser uma característica dos desenvolvedores destes softwares a grande preocupação com qualidade estrutural de software, o que pode levar a softwares com estas características. Além disso, a maior parte dos trabalhos têm investigado características no nível de sistema: os relacionamentos entre classes e objetos. Porém outras características interessantes em um nível de granularidade menor, como da classe e de métodos, devem ainda ser investigadas, por exemplo o grau de coesão interna dos módulos, grau de ocultação de informação e tamanho de métodos.

Dezenas de métricas têm sido propostas para avaliar diversos aspectos de software [15], em particular para a OO, destacam-se as métricas do conjunto MOOD [1] e do conjunto CK [3]. Tempero [13] avalia a necessidade de se conhecer os valores típicos destas métricas para que elas tenham uso efetivo na produção de software e destaca que muito pouco se sabe sobre isso. Baxter et al. [2] realizaram um estudo com o objetivo de investigar a estrutura de software desenvolvido em Java. Como os próprios autores salientam, este foi o primeiro trabalho a realizar um estudo desta natureza com um grande número de softwares. No es-

tudo de Tempero [13] foram utilizados 56 softwares abertos de tamanho pequeno a médio, e de diferentes domínios de aplicação. Os critérios para escolha dos softwares foram o fato de já terem sido utilizados em outros estudos ou serem amplamente utilizados, tais como tais como ArgoUML, Eclipse e NetBeans. Foram coletadas medidas das seguintes métricas para classes: número de métodos, campos, construtores, subclasses, interfaces implementadas, referências a uma classe como um membro, número de tipos dos campos, número de tipos usados como parâmetros nos métodos, referências à classe como parâmetro, referências à classe como retorno, número de tipos usados como retorno nos métodos, número de classes das quais se depende para compilar, número de classes que dependem da classe para compilar, número de métodos públicos; para interfaces: número de classes que as implementam; para método: número de instruções no *bytecode*, e para pacotes: número de classes no pacote. As medidas foram coletadas e analisadas para cada um dos softwares estudados. Os resultados desta análise mostram que algumas métricas têm distribuição que segue uma *power law* enquanto outras não. As métricas que têm distribuição *power law* incluem: referências a uma classe como um membro, referências à classe como parâmetro, referências à classe como retorno, número de subclasses. Dentre as métricas que não possuem distribuição *power law* estão: número de tipos dos campos, número de tipos usados como parâmetros nos métodos, número de tipos usados como retorno nos métodos. As métricas número de campos, número de métodos e número de métodos públicos também não seguem *power law*. As aplicações são de domínios diferentes e talvez domínios diferentes tenham distribuições diferentes; outra explicação é que os projetos das aplicações podem ser muito diferentes entre si.

O trabalho apresentado neste artigo avança o estudo das características de software aberto, avaliando métricas importantes como LCOM, COF e DIT, ainda não avaliadas desta forma em estudos precedentes. Detalhes dessas métricas são descritos na Seção III-A. Os softwares analisados neste estudo são de vários domínios de aplicação, o que representa melhor a diversidade de softwares existente. O investimento em qualidade estrutural é fator crítico de sucesso de softwares abertos, pois esse tipo de software é caracterizado por escassez de documentação e por grande quantidade de atividade de manutenção. Samoladas et al. [11] identificaram que a manutenibilidade de softwares abertos tende a ser melhor do que a de softwares fechados. Com base nisso, os resultados do estudo realizado neste trabalho possibilita identificar valores referência para as métricas estudadas. Esta é uma questão em aberto e sua solução constitui uma contribuição importante para viabilizar o emprego efetivo de métricas na produção de software.

III. METODOLOGIA

Os dados utilizados neste estudo são de 40 softwares abertos desenvolvidos em Java. Os softwares foram obtidos de *www.sourceforge.net*. Foram utilizados softwares de 11 domínios de aplicações diferentes, classificados pelo próprio *site*. Os dados relativos aos softwares e seus respectivos domínios são mostrados na Tabela I. No total foram analisadas 26.202 classes.

Foi utilizada a ferramenta *Connecta* [5] que coleta as medidas a partir do *byte code* das classes. Por esta razão, um dos critérios para a escolha dos softwares analisados foi a disponibilidade dos *bytecodes* das classes ou a facilidade de sua geração. Dentre os softwares analisados estão alguns de grande disseminação tais como Hibernate, *framework* Spring e a biblioteca JUnit, porém a escolha dos softwares não se baseou na sua popularidade, porque este critério poderia interferir nos resultados.

A. Métricas

Há uma grande quantidade de métricas de software orientado por objetos propostas na literatura. Dentre os trabalhos nesta linha, destacam-se o de Chidamber e Kemerer [3] e o de Abreu e Carapuça [1], nos quais são propostos os dois conjuntos de métricas para software OO mais referenciados na literatura, CK e MOOD, respectivamente. Neste trabalho serão utilizadas as seguintes métricas dos conjuntos CK e MOOD:

COF (*Coupling Factor* - Fator acoplamento): esta métrica do conjunto MOOD considera o conceito de relação *cliente-servidor* entre as classes constituintes de um software. Uma classe A é cliente de uma classe servidora B quando A referencia pelo menos um membro de B, seja este membro uma variável de instância ou um método. Uma relação cliente-servidor entre duas classes corresponde à existência de uma conexão entre elas. Em um software com n classes, o maior número possível de conexões é $n^2 - n$. A métrica 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. Um software totalmente conectado possui COF = 1. COF é uma métrica importante pois indica quão conectado é um software. Um software fortemente conectado possui estrutura rígida, baixo grau de independência entre os módulos e, consequentemente, um alto custo de manutenção.

LCOM (*Lack of Cohesion in Methods* - Ausência de coesão em métodos): é uma métrica do conjunto CK para a ausência de coesão entre os métodos de uma classe. Os autores desta métrica

Tabela I
SOFTWARES UTILIZADOS NO ESTUDO E SEUS RESPECTIVOS DOMÍNIOS

Domínio	*Software*	*#Classes*	*#Conexões*	*COF*
Clustering	Essence	182	543	0,016
	Gridsim	214	774	0,017
	JavaGroups	1061	3807	0,003
	Prevayler	90	137	0,017
	Super	246	1085	0,018
Database	DBUnit	289	911	0,011
	ERMaster	569	2187	0,007
	Hibernate	1359	5199	0,003
Desktop	Facilitator	2234	6565	0,001
	JavaGuiBuilder	60	126	0,036
	JavaX11Library	318	1146	0,011
	Jpilot	142	367	0,018
	Scope	214	535	0,012
Development	CodeGenerationLibrary	226	662	0,013
	DrJava	2766	9684	0,001
	FinfBugs	1019	3108	0,003
	JasperReports	1233	5610	0,004
	Junit	154	353	0,015
	SpringFramework	2116	7069	0,002
	BCEL	373	2111	0,015
Enterprise	Liferay	14	14	0,077
	Talend	2779	3567	0,000822
	Uengine	708	1774	0,004
	YAWL	382	1186	0,008
Financial	Jmoney	193	424	0,019
Games	JSpaceConquest	150	424	0,019
	Kolmafia	810	5106	0,008
	Robocode	29	16	0,02
Hardware	Jcapi	21	61	0,145
	LibUSBJava	35	90	0,076
	ServoMaster	55	117	0,039
Multimedia	CDK	3586	14711	0,001
	Jpedal	539	1533	0,005
	Pamguard	1503	5267	0,002
Networking	Bluecove	142	461	0,023
	DHCP	18	29	0,095
	JSLP	42	156	0,091
	WiKIDStrong Authentication	50	27	0,011
Security	JSCH	110	226	0,022
	OO Distributed Virtual Systems	171	325	0,011

consideram que a coesão entre os métodos de uma classe é definida pela similaridade entre eles. A avaliação da similaridade entre dois métodos é determinada pelo uso de variáveis de instância da classe em comum por eles. LCOM e dada pela diferença entre a quantidade de pares de métodos que não possuem variáveis de instância em comum e a quantidade de pares de métodos que possuem variáveis de instância em comum. Baixos valores para essa métrica indicam bom nível de coesão entre os métodos da classe avaliada.

DIT (*Depth of Inheritance Tree* - Profundidade da árvore de herança): indica a posição de uma classe na árvore de herança de um software, que é dada pela distância máxima da classe até a raiz da árvore. Essa métrica é considerada um indicador da complexidade de desenho e de predição do comportamento de uma classe, visto que quanto maior a profundidade da classe na árvore de herança, mais classes, e portanto mais métodos e atributos, estarão envolvidos na análise.

A métrica *Conexões Aferentes* [5] foi também analisada. Esta métrica é dada pelo total de classes que utilizam serviços de uma determinada classe. Do ponto de vista de manutenção, é importante conhecer as classes de maior impacto no sistema, ou seja, aquelas que têm grande número de conexões aferentes. Foram analisados também as quantidades de atributos e métodos públicos das classes.

B. Ajuste dos Dados

Para o ajuste dos dados foi utilizada a ferramenta Easy-Fit versão 5.0. Esta ferramenta realiza automaticamente o ajuste dos dados para diversas distribuições de probabilidades conhecidas, indicandos os melhores ajustes, além de realizar análises estatísticas básicas, tais como cálculo de média e desvio padrão. Dentre as distribuições discretas avaliadas pela ferramenta estão Bernoulli, Binomial, Uni-

forme, Geométrica, Hipergeométrica, Logarítmica, Binomial Negativa e Poisson; dentre as distribuições contínuas estão Beta, Uniforme, Normal, t-Student, Chi-quadrado, Erland, Exponencial, Frechet, Gama, Lognormal, Pareto e Weibull. Cada distribuição tem uma *pdf* (*probability density function*) e uma *cdf* (*cumulative distribution function*) característica. A *pdf* corresponde a uma equação que indica a probabilidade da variável aleatória assumir determinado valor *x*. A *cdf* corresponde a uma equação que indica a probabilidade da variável aleatória assumir um valor menor ou igual a *x*. As distribuições identificadas nos experimentos foram: Geométrica, Poisson e Weibull.

A distribuição Geométrica tem sua *pdf* , $f_g(x)$, e *cdf*, $F_g(x)$, com parâmetro p, dadas pelas Equações 2 e 3 respectivamente.

$$f_g(x) = P(X = x) = p(1-p)^{x-1}, 0 < p < 1 \quad (2)$$

$$F_g(x) = P(X \leq x) = 1 - (1-p)^{x}, 0 < p < 1 \quad (3)$$

A distribuição de Poisson é conhecida como a distribuição dos eventos raros, o que significa que a ocorrência de valores altos para a variável aleatória tem baixa probabilidade [12]. A distribuição de Poisson tem *pdf*, $f_p(x)$, e *cdf*, $F_p(x)$, definidas pelas Equações 4 e 5 respectivamente. O parâmetro λ da distribuição indica a média dos valores da variável aleatória.

$$f_p(x) = P(X = x) = \frac{e^{-\lambda}.\lambda^x}{x!} \quad (4)$$

$$F_p(x) = P(X \leq x_0) = \sum_{x=0}^{x=x_0} \frac{e^{-\lambda}.\lambda^x}{x!} \quad (5)$$

A distribuição Weibull tem sua *pdf*, $f_w(x)$, e sua *cdf*, $F_w(x)$, com parâmetros α e β, definidas pelas Equações 6 e 7 respectivamente. Essa distribuição aplica-se a situações em que a variável de interesse apresenta assimetria à esquerda, ou seja, quando há um número pequeno de ocorrências com valores altos para a variável de interesse e um número muito grande de ocorrências com valores baixos. Weibull é uma distribuição de cauda pesada. Conforme apresentado na Seção II, neste tipo de distribuição a média não é significativa.

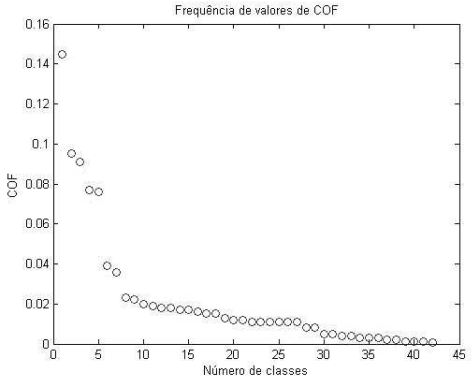
$$f_w(x) = P(X = x) = \frac{\alpha}{\beta}(\frac{x}{\beta})^{\alpha-1} e^{-(\frac{x}{\beta})^{\alpha}}, \alpha > 0 e \beta > 0 \quad (6)$$

$$F_w(x) = P(X \leq x) = 1 - e^{-(\frac{x}{\beta})^{\alpha}}, \alpha > 0 e \beta > 0 \quad (7)$$

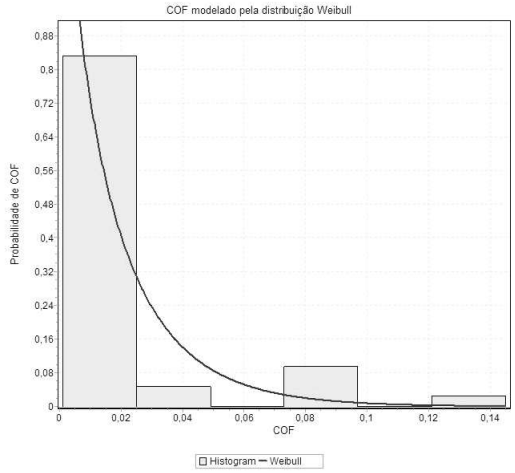
IV. Resultados

A. Métrica COF

O gráfico de dispersão de COF, mostrado na Figura 1, indica que valores menores do que 0.20 são muito mais frequentes do que valores maiores do que isso. Os valores de COF podem ser modelados pela distribuição Weibull, com parâmetros $\alpha = 0,91927$ e $\beta = 0,01762$. O gráfico da Figura 1 mostra o ajuste da distribuição aos dados coletados. Isso indica que a probabilidade de uma software possuir um valor alto para COF é baixa. Pela análise dos gráficos, mais de 80% dos softwares têm COF em torno de 0,02, a probabilidade de COF estar entre 0,02 e 0,14 é relativamente baixa, e a probabilidade de ser maior do que 0,14 tende a zero.



(a)



(b)

Figura 1. Dispersão de COF e seu ajuste à distribuição Weibull.

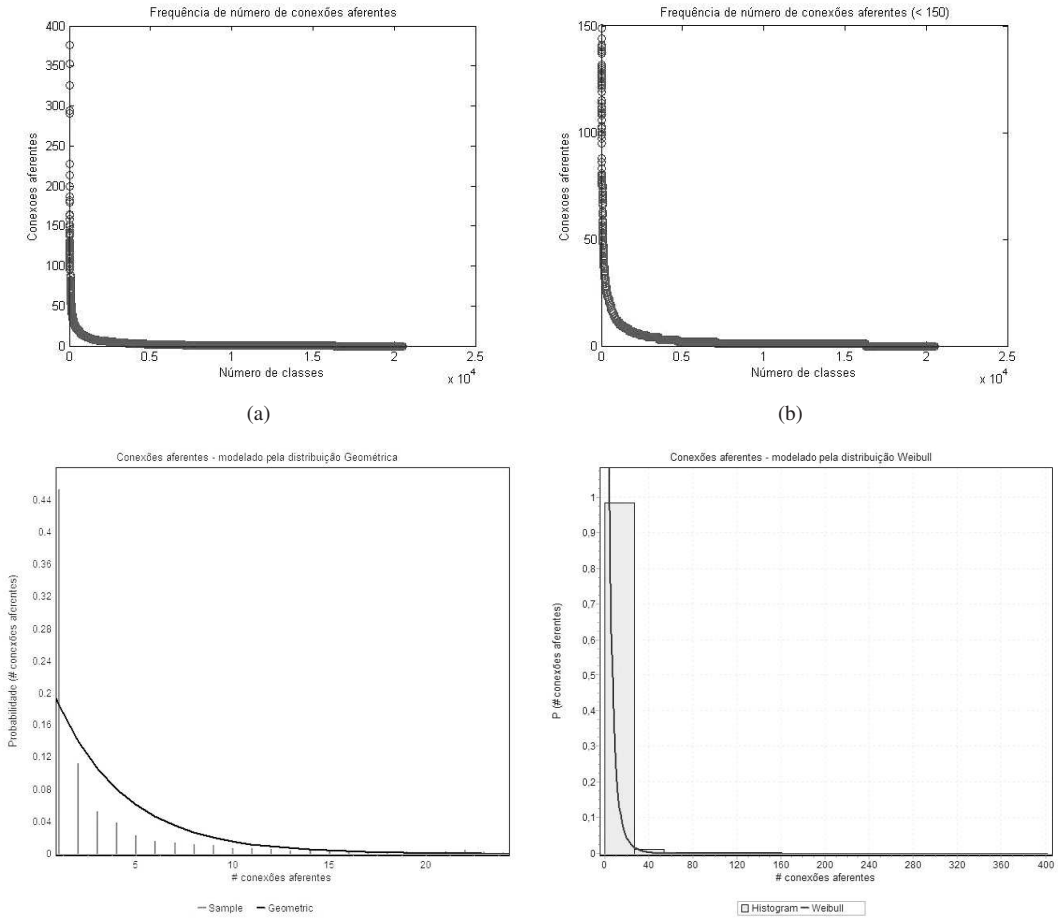


Figura 2. Conexões aferentes - gráficos de dispersão do conjunto completo e de valores menores do que 150. Ajustes às distribuições Geométrica e Weibull.

B. Métricas de Classe

Conexões Aferentes

A análise do gráfico de dispersão dos valores da métrica Conexões Aferentes, mostrado na Figura 2, sugere que se trata de uma distribuição de cauda pesada, com um número pequeno de classes com uma quantidade muito alta conexões aferentes e há uma quantidade muito grande de classes com poucas conexões aferentes. Isso indica que a maior parte das classes têm poucas classes dependentes. De fato, como mostra a Figura 2, os valores desta métrica podem ser modelados por duas distribuições: Geométrica e Weibull. A distribuição discreta Geométrica com parâmetro $p = 0,24248$ modela os dados desta métrica. Os dados podem também ser modelados pela distribuição contínua Weibull, com parâmetros $\alpha = 0,78986$ e $\beta = 3,2228$. A análise gráfica mostra que cerca de 50% da classes tem 1 conexão aferente. A probabilidade da medida desta métrica estar entre 1 e 20 é relativamente baixa, e de ser maior do que 20 tende a zero.

LCOM

A distribuição dos valores da métrica LCOM também é de cauda pesada. Os gráficos da Figura 3 mostram as dispersões do conjunto completo e de valores menores do que 10000. O ajuste realizado mostra que os valores desta métrica podem ser modelados pela distribuição Geométrica com parâmetro $p = 0,02645$, porém, a inspeção do gráfico, da Figura 3, mostra que esta distribuição não é perfeitamente adequada aos dados. Os dados podem também ser modelados pela distribuição de Weibull, com parâmetros $\alpha = 0,23802$ e $\beta = 1,465$. Esta distribuição mostra-se mais adequada aos dados. Pela análise gráfica da distribuição de valores desta métrica, cerca de 50% das classes têm LCOM igual a zero, o que indica bom nível de coesão. Encontram-se classes com LCOM entre 0 e 20 com uma frequencia relativamente baixa, menor do que 12%. A probabilidade de ocorrência de classes com LCOM maior do 20 tende a zero.

DIT

Os gráficos da Figura 4 mostram as dispersões dos

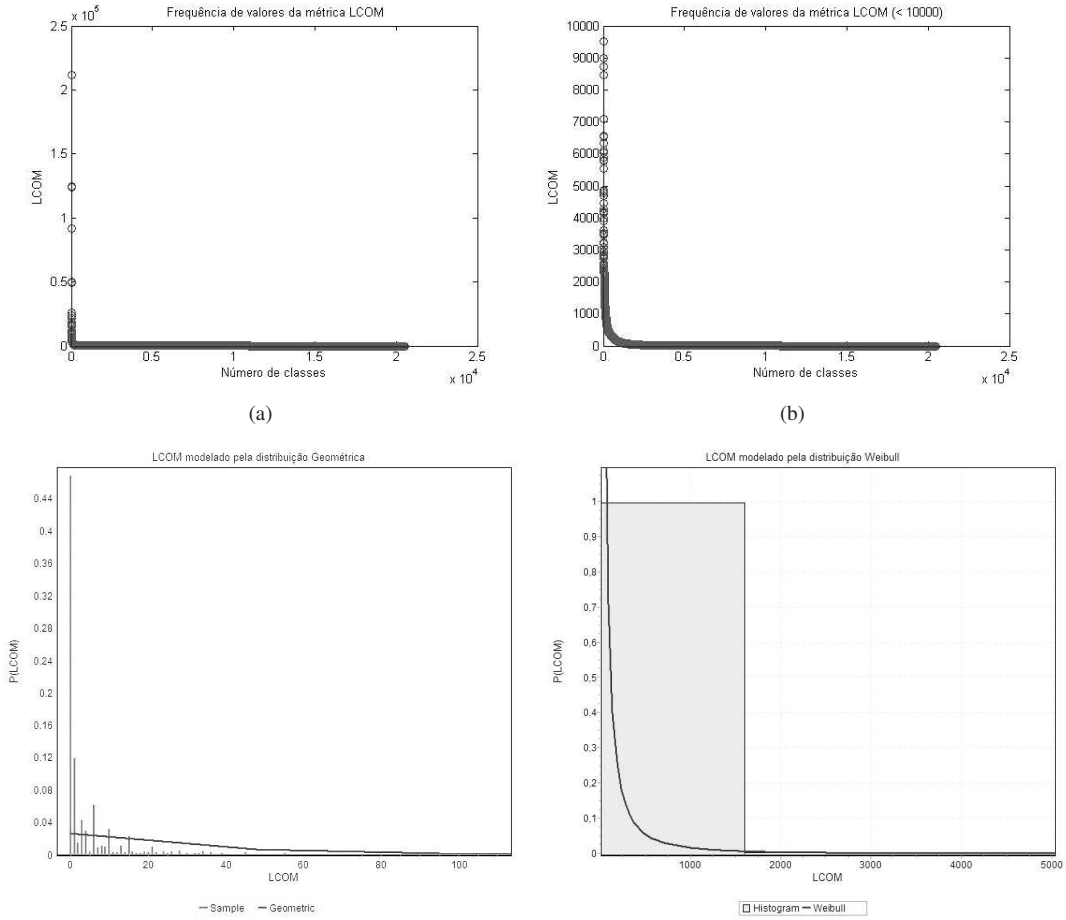


Figura 3. LCOM - gráfico de dispersão do conjunto completo e valores menores do que 10000. Ajustes às distribuições Geométrica e Weibull

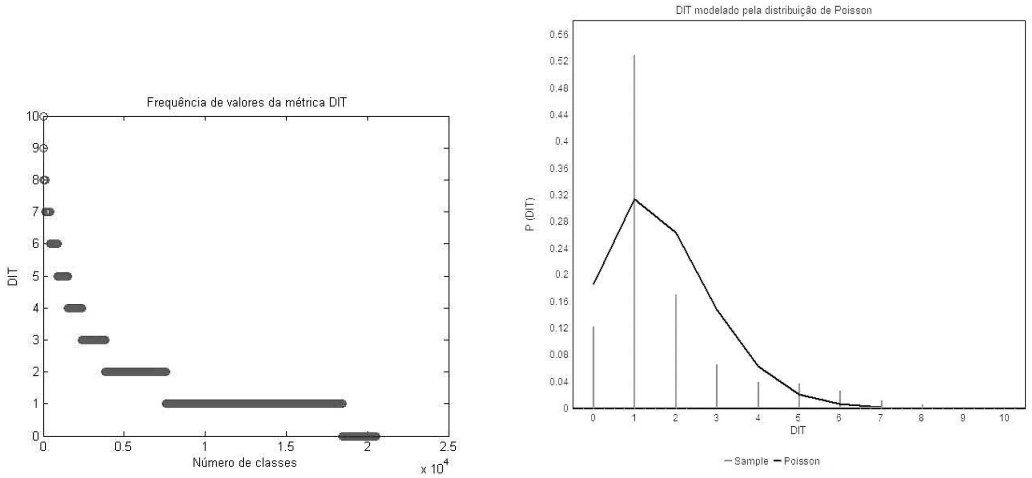
valores da métrica DIT e o ajuste à distribuição Poisson, com parâmetros $\lambda = 1,6818$ e $\beta = 3,2228$. Na distribuição de Poisson, o parâmetro $\lambda = 1,6818$ indica a média dos valores da variável aleatória. O valor referência sugerido pela análise dos dados para DIT é 2.

Atributos públicos

O gráfico da Figura 5 apresenta a dispersão da quantidade de atributos públicos de uma classe. A análise gráfica da dispersão mostra que há uma pequena quantidade de classes com um número muito alto de atributos públicos e a maior parte tem um número muito pequeno de atributos públicos, sendo que este número tende bruscamente a zero. O mesmo ocorre quando se consideram, por exemplo, apenas as classes que têm menos de 100 atributos públicos. Esta métrica pode ser modelada pela distribuição de Geométrica, com parâmetro $p = 0,48941$, e pela Weibull, com parâmetros $\alpha = 0,71008$ e $\beta = 4,4001$, como mostra a Figura 5. A análise gráfica mostra que mais de 75% das classes não possuem atributos públicos. A ocorrência de classes que possuem entre 1 e 8 atributos públicos é muito baixa, sendo menor do que 8%. A probabilidade de ocorrências de classes com mais de 8 atributos públicos tende a zero.

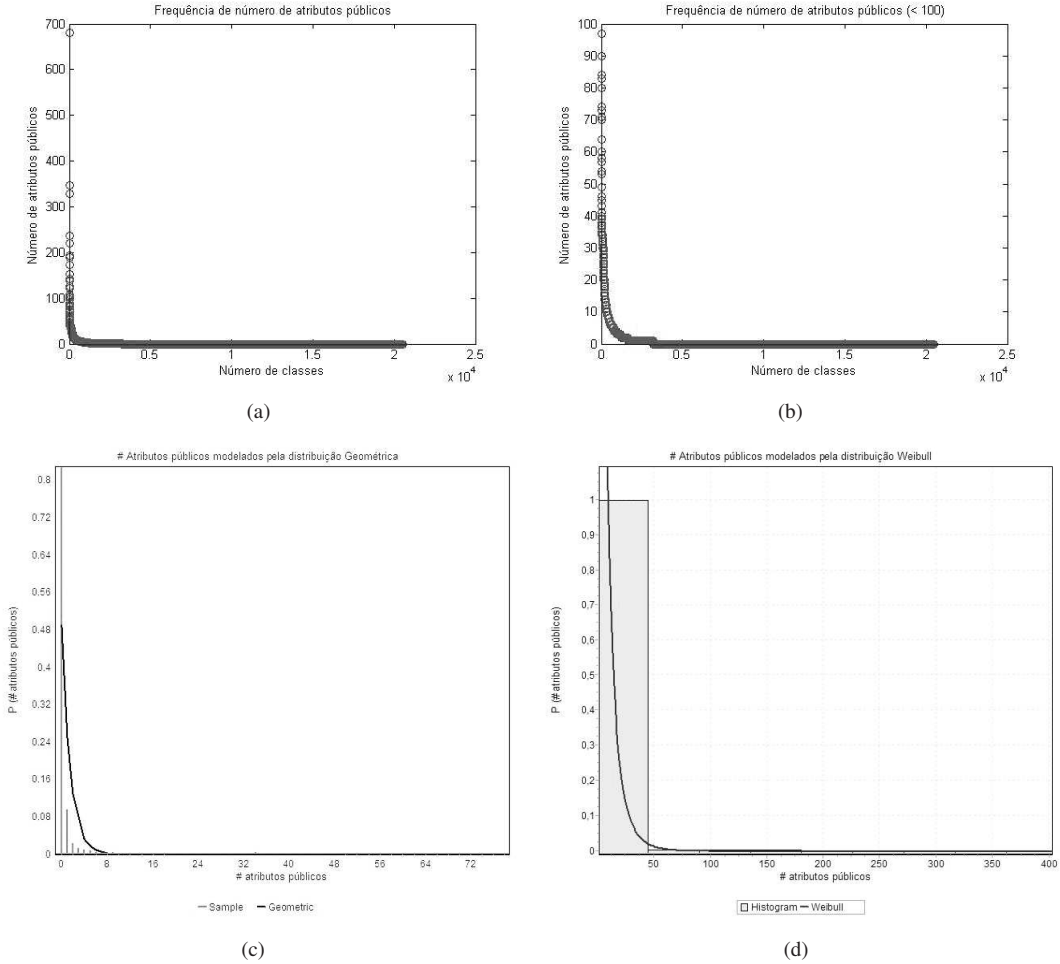
Métodos públicos

A dispersão da quantidade de métodos públicos de uma classe é mostrada no gráfico da Figura 6. Esta métrica pode ser modelada pela distribuição de Geométrica, com parâmetro $p = 0,14317$, e pela Weibull, com parâmetros $\alpha = 0,85938$ e $\beta = 5,6558$, como mostra a Figura 6. A análise gráfica da dispersão mostra que há uma pequena quantidade de classes com um número muito alto de métodos públicos e a maior parte tem um número muito pequeno de métodos públicos. O mesmo ocorre quando se consideram, por exemplo, apenas as classes que têm menos de 200 métodos públicos. A probabilidade de ocorrência de classes com mais de 40 métodos públicos tende a zero. Ocorrências de classe com número de métodos públicos entre 10 e 40 é relativamente baixa, e a maior parte das classes têm entre 0 e 10 métodos públicos.



(a) (b)

Figura 4. DIT - gráfico de dispersão e ajuste à distribuição de Poisson.



(a) (b)

(c) (d)

Figura 5. Atributos públicos - Gráfico de dispersão dos valores do conjunto completo e valores menores do que 100. Ajuste às distribuições Geométrica e Weibull.

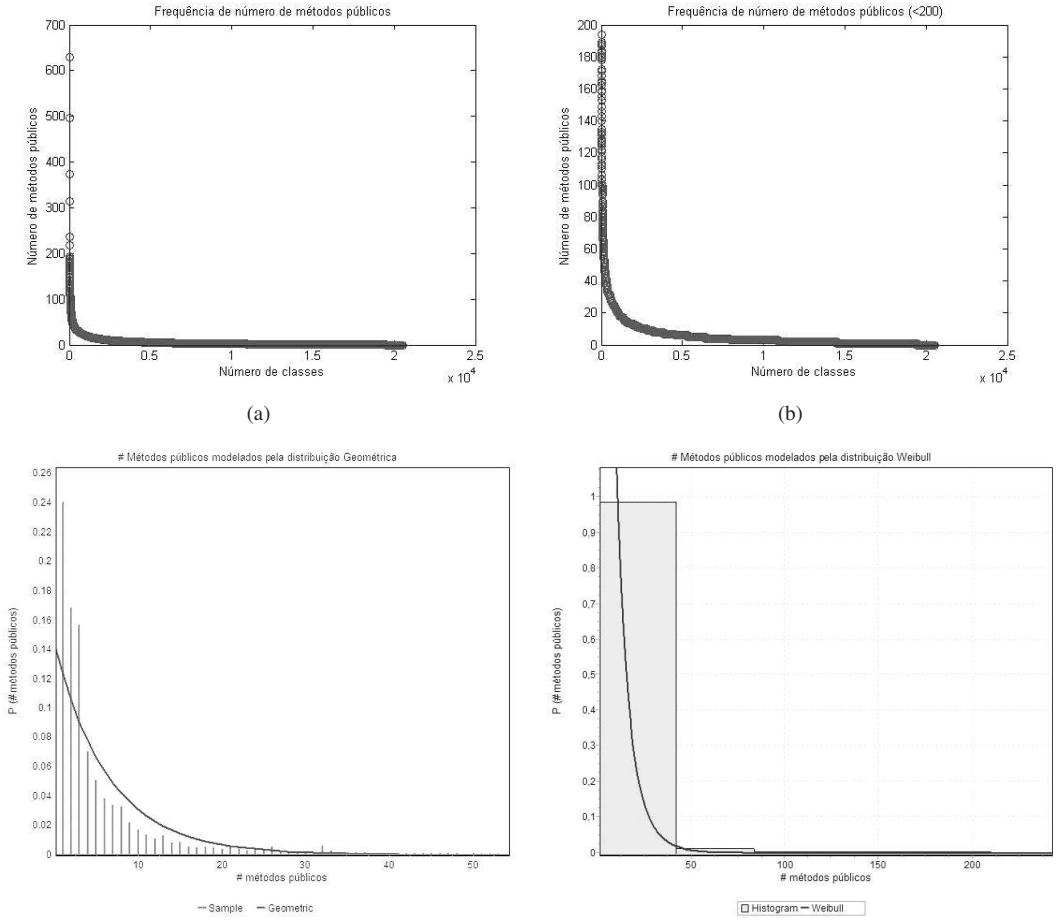


Figura 6. Métodos públicos - Gráfico de dispersão dos valores do conjunto completo e valores menores do que 200. Ajuste às distribuições Geométrica e Weibull.

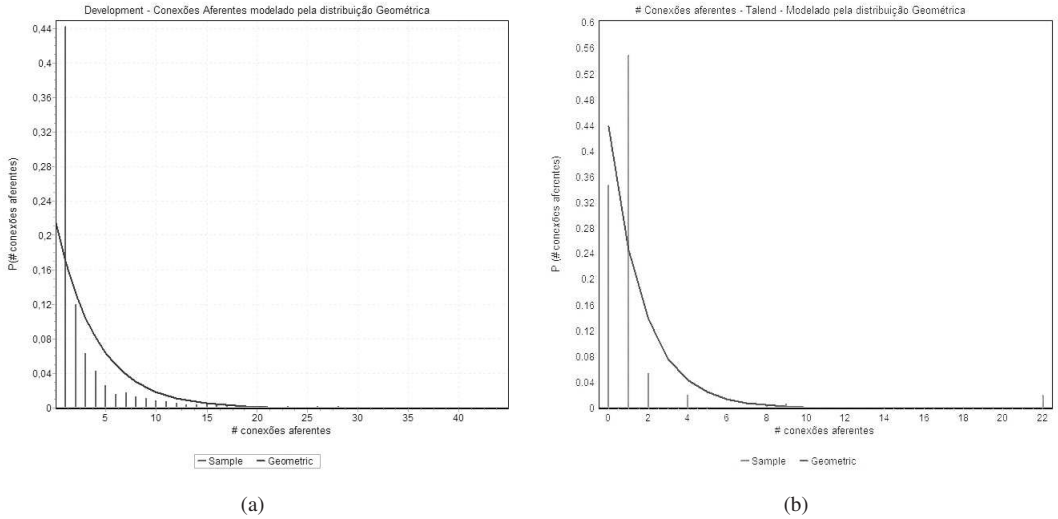


Figura 7. Conexões aferentes - Domínio *Development* e Talend. Modelagem pela distribuição Geométrica.

C. Ajuste para os diversos domínios e para um software

O ajuste dos dados para os diversos domínios de aplicação mostrou que as mesmas distribuições de probabilidades encontradas para as métricas no experimento com o conjunto de dados completo são aplicáveis. Como são 11 domínios de aplicação e 5 métricas, e a análise foi realizada com distribuições discretas e contínuas, o que resulta em um número grande de gráficos, a título de exemplificação é mostrado o gráfico de ajuste da métrica Conexões aferentes para o domínio *Development* e para o software Talend na Figura 7.

D. Análise dos Resultados

A análise da distribuição dos valores da métrica COF mostra que não há um valor típico para esta métrica, e que a maior parte dos softwares OO abertos tem baixo grau de conectividade. COF é modelada pela distribuição Weibull. De acordo com o valores observados, sugere-se adotar os seguintes valores referência para COF: até 0,02 (bom), entre 0,02 e 0,14 (regular) e maior ou igual a 0,14 (ruim).

A métrica conexões aferentes indica quantas classes dependem da classe analisada. Esse valor é de grande importância para se avaliar o cuidado que se deve tomar ao alterar a classe. A distribuição de valores desta métrica mostra que para um determinada classe há um número pequeno de classes que dependem dela.

Não há um valor típico para esta métrica. A partir dos resultados obtidos, sugere-se adotar os seguintes valores como referência: 1 (bom), entre 1 e 20 (regular) e maior ou igual a 20 (ruim).

LCOM também pode ser modelada pela distribuição de Weibull. Há poucos softwares com valores altos de LCOM, ou seja, há poucas classes com baixa coesão. A inspeção gráfica da distribuição de probabilidade desta métrica mostra que cerca de 50% das classes tem LCOM igual a 0, o que corresponde a um bom nível de coesão. Sugere-se adotar a seguinte escala como referência para LCOM: 0 (bom), entre 0 e 20 (regular) e maior ou igual a 20 (ruim).

A distribuição de Poisson modela os valores das métricas DIT dos softwares OO desenvolvidos em Java. Desta forma, há uma probabilidade muito pequena de ocorrer valores altos para esta métrica. A média de DIT na amostra avaliada é 1,68, aproximadamente 2.

Esse valor pode ser tomado como referência para a construção de software OO, ou seja, pode-se considerar que é desejável que uma classe esteja no máximo no terceiro nível na árvore de herança.

As quantidades de atributos públicos de métodos públicos podem ser modeladas pela distribuição Geométrica e pela Weibull, o que indica que há poucas classes com número alto de atributos públicos e poucas classes com um número alto de métodos públicos. Isso indica que as classes seguem os princípios de ocultação de informação e têm interface pequena, ou seja, exportam poucos serviços. Isso é condizente com o fato de terem alta coesão interna, mostrado pelos baixos valores da métrica LCOM. A análise gráfica da distribuição das quantidade de atributos públicos mostra que mais de 75% das classes não possuem atributos públicos. Isso já tem sido uma diretriz para a construção de software OO. Pelos dados obtidos, sugerem-se os seguintes valores referência para a quantidade de atributos públicos de uma classe: 0 (bom), entre 0 e 8 (regular) e maior ou igual a 8 (ruim). A análise gráfica da distribuição de valores de métodos públicos mostra que a probabilidade de uma classe possuir mais de 40 métodos aproxima-se de zero. Os seguintes valores referência são sugeridos para essa métrica: de 0 a 10 (bom), entre 10 e 40 (regular) e maior ou igual a 40 (ruim).

Os valores sugeridos como referência para as métricas COF, LCOM, DIT, conexões aferentes, quantidade de atributos públicos e de métodos públicos obtidos a partir da análise dos resultados do experimento são sumarizados na Tabela II.

V. Conclusão

Este trabalho apresenta um estudo sobre caracterização de software OO abertos. No estudo foram utilizados 40 softwares desenvolvidos em Java de 11 domínios de aplicação diferentes, em um total de 26.202 classes. Investigou-se se as métricas COF (conectividadade), LCOM (coesão), DIT (profundidade da árvore de herança), quantidade de conexões aferentes, quantidade de atributos públicos e quantidade de métodos públicos podem ser modeladas por uma distribuição de probabilidade. O estudo concluiu que a distribuição de Weibull modela os valores de: COF, o que indica que poucos softwares abertos apresentam alto grau de conectividade; conexões aferentes, o que indica que há poucas classes com um número alto de classes dependentes; LCOM, o que mostra que a maior

Tabela II
Valores referência para métricas OO

Fator	*Nível*	*Métrica*	*Valor Referência*
Conectividade	Sistema	COF	Bom: até 0,02 - Regular: entre 0,02 e 0,14 - Ruim: maior ou igual a 0,14
	Classe	# conexões aferentes	Bom: 1 - Regular: entre 1 e 20 - Ruim: maior ou igual a 20
Ocultação de Informação	Classe	# atributos públicos	Bom: 0 - Regular: entre 0 e 8 - Ruim: maior ou igual a 8
Tamanho da Interface	Classe	# métodos públicos	Bom: 0 a 10 - Regular: entre 10 e 40 - Ruim: maior ou igual a 40
Herança	Classe	DIT (profundidade na árvore de herança)	Valor típico: 2
Coesão Interna	Classe	LCOM (ausência de coesão)	Bom : 0 - Regular: entre 0 e 20 - Ruim: maior ou igual a 20

parte das classes tem bom nível de coesão; quantidade de atributos públicos, o que evidencia que a ocultação de informação tem sido devidamente empregada; quantidade de métodos públicos, o que informa que a quantidade de serviços exportados por uma classe tende a ser pequena. A distribuição de Poisson modela os valores da métrica DIT, o que torna possível identificar um valor típico para essa métrica.

Com base nos resultados dos experimentos deste trabalho propõe-se utilizar os valores das métricas identificados nos softwares abertos como referência para a construção de software de qualidade. Este é um dos primeiros trabalhos na direção de se identificar esses valores. Isso é uma questão de grande relevância para o uso efetivo de métricas na construção de software. Neste trabalho foram investigados os valores de seis importantes métricas de software OO. Sugere-se utilizar o método aqui empregado para o estudo de outras métricas.

AGRADECIMENTOS

Os autores agradecem à Fapemig, entidade financiadora do projeto CONNECTA - Conectividade em Módulos (Processo: CEX APQ-3999-5.01/07, Fapemig - Edital Universal), no qual foi realizado o trabalho apresentado neste artigo.

REFERÊNCIAS

[1] Abreu, Fernando Brito; Carapuça, Rogério. (1994). *Object-Oriented Software Engineering: Measuring and Controlling the Development Process.* In: Proceedings of 4th Int. Conf. of Software Quality, McLean, VA, USA, 3-5 October 1994.

[2] Baxter, G; Frean, M.; Noble, J; Rickerby, M; Smith, H; Visser, M; Melton, H; Tempero, E. (2006) *Undertanding the Shape of Java Software*. In: OOPSLA'06. Oregon, Portland, USA, 22-26 October 2006.

[3] Chidamber, Shyam R.; Kemerer, C.F. (1994) *A Metrics Suite for Object Oriented Design*. IEEE Transactions on Software Engineering, pp. 476-493, 1994.

[4] Fenton, Norman; NEIL, Martin. (2000) *Software Metrics: Roadmap*. In: Proceedings of the Conference on the Future of Software Engineering, Maio de 2000.

[5] Ferreira, Kecia A. M.; Bigonha, Mariza. A. S.; Bigonha, Roberto. S.(2008) *Reestruturação de Software Dirigida por Conectividade para Redução de Custo de Manutenção*. Revista de Informática Teórica e Aplicada. Vol. 15. No 2. Rio Grande do Sul: 2008. pp: 155-179.

[6] Louridas, P.; Spinellis, D.; Vlachos, V. (2008) *Power Laws in Software*. ACM Transactions on Software Engineering and Methodology, Vol. 18, No 1, Article 2. Setembro de 2008.

[7] Martin, Robert.(1994) *OO Design Quality Metrics - An analysis of dependencies*. Disponível em http://www.objectmentor.com/resources/articles/oodmetrc.pdf. Acesso de Julho de 2009.

[8] Newman, M. E. J.(2003) *The structure and function of complex networks*. SIAM Reviews, Vol. 45. No 2, pp: 167-256.

[9] Potantin, A.; Noble, J.; Frean, M.; Biddle, R.. (2005) *Scale-Free Geometry in OO Programs*. Communications of the ACM. May 2005. Vol. 48. Nº 5. pp. 99-103.

[10] Puppin, D.; Silvestrini, F.. (2006) *The Social Network of Java Classes*. SAC'06. Abril de 2006. Dijon, França. pp. 1409-1413.

[11] Samoladas, I; Stamelos, I; Angelis, L; Oikonomou, A. (2004) *Open Source Software Development Should Strive for Even Greater Code Maintainability*. Communications of the ACM. October 2004/Vol. 47. no 10.pp. 83-87.

[12] Soares, J. F.; Farias, A. A.; Cesar, C. C. (1991) *Introdução à Estatística*. Rio de Janeiro: Ed. Guanabara, 1991.

[13] Tempero, Ewan. (2008) *On Measuring Java Software*. In: ACSC2008. Wollongong, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 74. 2008.

[14] Valverde, S.; Ferrer-Cancho, R.; Sole, R.. (2002) *Scale-free networks from optimal design.* Europhysics Letters 60, 4. Novembro de 2002. pp 512-517.

[15] Xenos, M.; Stavrinoudis, D.; Zikouli, K.; Christodoulakis, D. (2000) *Object-Oriented Metrics - A Survey*. Proceedings of the FESMA 2000, Madrid, Spain, 2000.

[16] Wheelson, R.; Counsell, S.. (2003) *Power law distributions in class relationships.* In: Proceedings of 3rd International Workshop on Source Code Analysis and Manipulation (SCAM), Amsterdam, Setembro de 2003.