Modelos Estocásticos para Síntese de Sistemas de Software Organizados em Módulos *DRAFT*

Rodrigo Rocha Gomes e Souza

ABSTRACT

Existem muitos algoritmos para identificar os módulos de um sistema a partir de sua implementação, mas há poucos estudos empíricos sobre esses algoritmos. Isso ocorre porque é difícil encontrar sistemas cuja arquitetura modular é bem documentada. Propomos suprir essa escassez através da geração automática de sistemas de *software* sintéticos, viabilizando análises empíricas em larga escala. Neste artigo apresentamos e avaliamos três modelos estocásticos que cumprem esse papel.

1. INTRODUÇÃO

A divisão conceitual de um sistema de *software* em módulos é muito importante no gerenciamento de sua evolução [11]. À medida que os desenvolvedores de um sistema são substituídos, no entanto, essa informação pode se perder. Alguns pesquisadores propõem o uso de algoritmos de aglomeração (*clustering*) para recuperar a arquitetura modular de um sistema a partir de sua implementação [8, 2, 9].

Espera-se que os algoritmos encontrem organizações modulares semelhantes àquelas que seriam encontradas por especialistas nos sistemas analisados. Uma forma de validar os algoritmos consiste, pois, em aplicá-los ao código-fonte de um sistema cuja organização em módulos esteja documentada e então comparar, através de uma medida de similaridade entre particionamentos [12, 14], os módulos do sistema com os módulos encontrados pelos algoritmos. Infelizmente é difícil encontrar sistemas documentados nesse nível de detalhe e por essa razão a avaliação empírica dos algoritmos de recuperação de arquitetura ainda é incipiente.

Para suprir a escassez de sistemas bem documentados, propomos o uso de sistemas de *software* sintéticos. Esses sistemas são gerados por computador a partir de modelos parametrizáveis e obedecem a estruturas modulares conhecidas. Com essa abordagem é possível submeter os algoritmos a amostras potencialmente infinitas de sistemas arbitrariamente complexos e, ainda, estudar como os parâmetros dos modelos afetam a avaliação dos algoritmos.

Na próxima seção apresentamos três modelos estocásticos que podem ser usados para gerar sistemas sintéticos. Na seção 3 descrevemos um experimento para avaliar os modelos quanto à capacidade de gerar sistemas semelhantes a sistemas reais. Os resultados do experimento são apresentados na seção 4. Na seção 5 discutimos os resultados e os trabalhos futuros.

2. MODELOS ESTOCÁSTICOS

A implementação de um sistema possui muitos detalhes, mas nem todos os detalhes são relevantes para se compreender sua organização modular. Para recuperar os módulos de um sistema orientado a objetos é comum considerar apenas as suas classes e os relacionamentos de dependência entre elas — uma rede de dependências entre classes. Essas redes podem ser extraídas automaticamente a partir da análise estática do código-fonte ou do código objeto do sistema que se deseja estudar. (Neste estudo consideramos apenas sistemas orientados a objetos, mas os conceitos podem ser igualmente aplicados a outros paradigmas de programação.)

Pesquisas recentes mostram que redes de dependências entre classes possuem propriedades estatísticas comuns a redes complexas estudadas em diversos domínios, tais como sociologia, biologia e linguística [10, 13]. Essas propriedades foram incorporadas a diversos modelos de geração de redes [3, 7] que podem ser usados para gerar redes de dependências entre classes.

2.1 O Modelo LFR

Lancichinetti, Fortunato e Radicchi [7] criaram um modelo de rede com estrutura de módulos embutida baseado em distribuições estatísticas encontradas em redes de vários domínios [1]. Ele gera grafos não-orientados e não-ponderados cuja distribuição dos graus dos vértices e cuja distribuição dos tamanhos dos módulos são ambas leis de potência. Mais precisamente, o número de vértices cujo grau é k é proporcional a $k^{-\gamma}$, onde o expoente γ tipicamente varia entre 2 e 3, e o número de módulos com n vértices é proporcional a $n^{-\beta}$, com β variando entre 1 e 2.

O modelo possui os seguintes parâmetros:

[1]Uma implementação desse modelo está disponível em `http://santo.fortunato.googlepages.com/benchmark_2_2.tar`

- número de vértices, $|V|$;
- expoente da distribuição de graus, γ;
- grau médio, $\langle k \rangle$;
- grau máximo, k_{max};
- expoente da distribuição de tamanhos dos módulos, β;
- número mínimo de vértices por módulo, T_{min};
- número máximo de vértices por módulo, T_{max};
- parâmetro de mistura, μ.

O modelo produz redes que seguem aproximadamente os parâmetros fornecidos. O parâmetro de mistura indica a proporção de arestas externas em relação ao total de arestas de cada vértice. Arestas externas são aquelas que ligam vértices de módulos distintos. Assim, se $\mu = 0,4$ e o vértice v pertence ao módulo M, então 40% das suas arestas estão ligadas a vértices que não pertencem ao módulo M.

O modelo gera grafos não-orientados, uma representação muito simplificada das redes de dependências entre classes. Além disso, ele considera que todos os vértices possuem arestas externas, o que certamente não é verdade no domínio de *software*. Por fim, os vértices de um módulo podem se ligar a vértices de qualquer outro módulo, sem restrição. Essa característica difere do que se encontra em programas de computador, onde as dependências entre módulos são controladas (por exemplo, ao se adotar uma arquitetura em camadas).

2.2 O Modelo BCR+

Bollobás, Borgs, Chayes e Riordan criaram um modelo de rede orientada, o modelo BCR, inspirado na rede de *links* entre páginas da *web* [3]. Considerando as limitações do modelo LFR, propomos o modelo BCR+, que é extensão do modelo BCR que considera a organização dos vértices em módulos[2]. O modelo possui os seguinte parâmetros:

- número de vértices, $|V|$;
- arquitetura modular, A;
- probabilidades p, q e r, tal que $p + q + r = 1$;
- probabilidade μ;
- δ_{in} e δ_{out}.

A arquitetura modular é uma rede que representa as dependências permitidas entre módulos. Dois vértices da rede gerada pelo modelo só podem estar ligados se os módulos correspondentes na arquitetura estiverem ligados por uma aresta.

[2]Uma implementação pode ser baixada em `http://code.google.com/p/swasr/wiki/Instalacao` TODO: criar empacotamento e página só dos modelos

O modelo é implementado por um algoritmo que inicialmente cria um vértice com autolaço (aresta ligando um vértice a ele próprio) para cada módulo representado na arquitetura; cada vértice é incluído no módulo correspondente. A seguir são efetuadas alterações sucessivas à rede até que ela contenha $|V|$ vértices.

Na descrição do algoritmo apresentada a seguir, "escolher um vértice v de acordo com $k_{out} + \delta_{out}$" significa escolher um vértice v de modo que a probabilidade de se escolher um vértice v_i é proporcional a $k_{out}(v_i) + \delta_{out}$, onde $k_{out}(v_i)$ é o grau de saída do vértice v_i; analogamente, k_{in} significa grau de entrada. Cada iteração do algoritmo realiza uma das seguintes alterações:

- **Criação de vértice com grau de saída = 1**. Com probabilidade p é criado um vértice v juntamente com uma aresta de v para um vértice pré-existente w, onde w é escolhido de acordo com $k_{in} + \delta_{in}$. O vértice v é incluído no módulo de w.
- **Criação de vértice com grau de entrada = 1**. Com probabilidade q é criado um vértice w juntamente com uma aresta de um vértice pré-existente v para w, onde v é escolhido de acordo com $k_{out} + \delta_{out}$. O vértice w é incluído no módulo de v.
- **Criação de uma aresta**. Com probabilidade r é criada uma aresta de um vértice pré-existente v para um vértice pré-existente w, v escolhido de acordo com $k_{out} + \delta_{out}$ e w escolhido de acordo com $k_{in} + \delta_{in}$. A escolha de w não é livre: com probabilidade $1 - \mu$, w é escolhido dentre os vértices do qual v faz parte; com probabilidade μ, w é escolhido dentre os vértices de módulos adjacentes ao módulo de v segundo a arquitetura.

Como neste modelo a rede é orientada, pode-se considerar separadamente uma distribuição dos graus de entrada e uma distribuição dos graus de saída. Da mesma forma que no modelo LFR, essas distribuições seguem leis de potência, com expoentes γ_{in} e γ_{out}, respectivamente. Pode-se demonstrar analiticamente que, quando $|V|$ tende a infinito, os expoentes podem ser calculados pelas seguintes expressões [3]:

$$\gamma_{in} = 1 + \frac{1 + \delta_{in}(p + q)}{p + r} \tag{1}$$

$$\gamma_{out} = 1 + \frac{r + q}{1 + \delta_{out}(p + q)} \tag{2}$$

Este modelo supera as limitações encontradas no modelo LFR: as redes são orientadas, são permitidos vértices sem arestas externas e é possível restringir as dependências entre módulos através da arquitetura fornecida como parâmetro. Além disso ele é um modelo evolutivo: o algoritmo pode ter como ponto de partida uma rede existente e então expandi-la criando mais vértices e arestas. A desvantagem em relação ao modelo LFR é o controle reduzido sobre a rede:

não há como impor restrições sobre o grau máximo, sobre a distribuição dos tamanhos dos módulos e nem estabelecer limites de tamanho para os módulos.

3. EXPERIMENTO

Na seção anterior discutimos características gerais dos modelos de geração de redes. Com o experimento, procuramos responder à seguinte pergunta: os modelos são capazes de produzir redes sintéticas parecidas com redes extraídas de sistemas reais?

TODO: desenvolver algo que está nos comentários do arquivo fonte

O experimento foi dividido em cinco etapas: extração das redes dos sistemas (redes reais), análise do sistema de referência, escolha dos parâmetros dos modelos, geração de redes sintéticas e comparação entre as redes.

CONSIDERAMOS REDES NÃO-ORIENTADAS.

3.1 Extração das Redes dos Sistemas

Os sistemas analisados foram implementados na linguagem Java e distribuídos em diversos arquivos JAR, cada arquivo contendo várias classes. Para construir a rede de um sistema, consideramos que cada arquivo JAR é um módulo arquitetural. Essa é uma aproximação razoável, uma vez que arquivos JAR distintos são, em geral, desenvolvidos por equipes diferentes. A extração das dependências entre as classes foi realizada pela ferramenta DepFind[3].

3.2 Análise das Redes Reais

A partir da análise da rede de referência que foi extraída do sistema VilloNanny, as seguintes métricas foram computadas:

- número de vértices, $|V|$;
- número de arestas, $|E|$;
- número de arestas externas $|E_{ext}|$;
- grau médio, $\langle k \rangle$;
- grau máximo, k_{max};
- expoente da distribuição de graus, γ;
- número de módulos, $|M|$;
- tamanho do menor módulo, t_{min};
- tamanho do maior módulo, t_{max};
- expoente da distribuição dos tamanhos dos módulos, β

Essas métricas são muito usadas para caracterizar redes complexas. Neste experimento as métricas foram usadas para ajustar os parâmetros dos modelos.

[3] `http://depfind.sourceforge.net/`

Os expoentes das distribuições de graus e de tamanhos dos módulos foram estimados através da técnica de máxima verossimilhança [5]. Utilizamos uma implementação disponível na Internet[4].

Além disso, extraímos a arquitetura modular do sistema. Na arquitetura, existe uma aresta de um módulo A para um módulo B somente se pelo menos um vértice de A se liga a um vértice de B.

3.3 Escolha dos Parâmetros dos Modelos

Queremos gerar redes com características próximas às características da rede de referência, e isso depende de uma escolha adequada dos parâmetros dos modelos. Na discussão a seguir, X^s e X^m representam, respectivamente, o valor de uma métrica X extraída da rede de referência e o valor de um parâmetro X de um modelo.

No modelo LFR os parâmetros coincidem com as métricas extraídas da rede de referência, com exceção do parâmetro de mistura, μ, que foi calculado de acordo com a expressão $|E_{ext}|^s \ / \ |E|^s$.

TODO: definir melhor, abaixo, a escolha de parâmetros do modelo BCR+

Para o modelo BCR+ a arquitetura modular fornecida como parâmetro é igual à arquitetura extraída do sistema, número de vértices idem e calculamos $p, r, q, \delta_{in}, \delta_{out}$ de forma a satisfazer às seguintes restrições:

- $q = p$. Como no estudo consideramos redes não-orientadas, não faz sentido distinguir os dois casos em que há a criação de um vértice e uma aresta, já que apenas o sentido da aresta é alterado.
- O número de arestas do modelo deveria ser igual ao número de arestas do sistema
- O grau máximo ...
- O tamanho da maior comunidade ...

3.4 Geração de Sistemas Sintéticos

Considerando os parâmetros escolhidos, geramos 9 redes a partir de cada um dos modelos.

3.5 Comparação Entre Sistemas Sintéticos e Sistemas Reais

Devido à escolha de parâmetros, as redes sintéticas compartilham os valores de diversas métricas com as redes reais correspondentes, tais como número de vértices e número de arestas. Essas métricas, no entanto, representam apenas características globais de uma rede, e nada dizem sobre a organização local dos vértices e arestas. Por essa razão, a comparação entre as redes sintéticas e as redes reais foi realizada através de uma métrica de distância entre redes de mesmo tamanho [1]. Quanto menor a distância entre duas redes, maior a semelhança entre elas.

TODO: descrever a métrica.

[4] `http://www.santafe.edu/~aaronc/powerlaws/`

Table 1: Métricas do sistema VilloNanny

$\|V\|$	$\|E\|$	$\|E_{ext}\|$	$\|M\|$	γ	β
1658	6766	343	13	2.68	1.00

Table 2: Discrepâncias relativas

	$\|E\|$	$\|E_{ext}\|$	$\|M\|$	γ	β
LFR	0.44%	0.06%	27.35%	4.10%	4.44%
BCR+	2.45%	2.17%	0.00%	19.69%	1.67%

TODO: dizer que consideramos redes não-orientadas.

TODO: distância entre os dois sistemas.

4. RESULTADOS

5. DISCUSSÃO

Este artigo apresentou uma nova abordagem, baseada na geração de sistemas sintéticos, de avaliação de algoritmos de aglomeração aplicados à recuperação de módulos de sistemas de *software*. Três modelos de sistemas sintéticos foram apresentados e avaliados experimentalmente quanto à capacidade de produzir sistemas realistas.

O experimentou revelou que

ESPECULAÇÃO SOBRE O PAPEL DA MODELAGEM ESTATÍSTICA NA ENGENHARIA DE SOFTWARE...

Acreditamos que outras linhas de pesquisa da engenharia de *software*, a exemplo da análise de impacto e da localização de funcionalidades, podem se beneficiar do tipo de avaliação apresentado neste artigo. Para isso é preciso ampliar os modelos aqui discutidos com os conceitos específicos de cada linha de pesquisa.

Trabalhos futuros: explorar métricas de arquitetura, avaliar algoritmos de clustering, refinar modelos / considerar outros modelos.

Outros modelos a considerar: [6, 4].

Table 3: Distâncias em relação à rede do VilloNanny

Modelo/Sistema	Distância Média	Distância Mínima
LFR	$3,28 \pm 0,01$	$3,26$
BCR+	$2,60 \pm 0,28$	$2,16$
JFreeChart	$2,44 \pm 0,00$	$2,44$

6. REFERENCES

[1] R. F. S. Andrade, J. G. V. Miranda, S. T. R. Pinho, and T. P. Lobão. Measuring distances between complex networks. *Physics Letters A*, 372(32):5265–5269, August 2008.

[2] P. Andritsos and V. Tzerpos. Software clustering based on information loss minimization. In *Proc. 10th Working Conference on Reverse Engineering WCRE 2003*, pages 334–344, 2003.

[3] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan. Directed scale-free graphs. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 132–139, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[4] T. Chen, Q. Gu, S. Wang, X. Chen, and D. Chen. Module-based large-scale software evolution based on complex networks. *8th IEEE International Conference on Computer and Information Technology*, pages 798—803, 2008.

[5] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data, 2007.

[6] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. 2009.

[7] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 78(4), 2008.

[8] S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen, and E. R. Gansner. Using automatic clustering to produce high-level system organizations of source code. In *Proc. th International Workshop on Program Comprehension IWPC '98*, pages 45–52, 1998. Bunch.

[9] O. Maqbool and H. A. Babri. Hierarchical clustering for software architecture recovery. 33(11):759–780, 2007.

[10] C. R. Myers. Software systems as complex networks: structure, function, and evolvability of software collaboration graphs. *Phys Rev E Stat Nonlin Soft Matter Phys*, 68(4 Pt 2):046116, Oct 2003.

[11] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12):1053–1058, 1972.

[12] V. Tzerpos and R. C. Holt. Mojo: a distance metric for software clusterings. In *Proc. Sixth Working Conference on Reverse Engineering*, pages 187–193, 1999. MoJo.

[13] S. Valverde and R. V. Solé. Hierarchical small worlds in software architecture. (Directed Scale-Free Graphs), 2003.

[14] Z. Wen and V. Tzerpos. Evaluating similarity measures for software decompositions. In *Proc. 20th IEEE International Conference on Software Maintenance*, pages 368–377, 2004. EdgeMoJo.