Modelos Estocásticos para Síntese de Sistemas de Software Organizados em Módulos
DRAFT v2

Rodrigo Rocha Gomes e Souza

22 de maio de 2009

Resumo

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 dois 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 organização 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.

O sucesso dessa abordagem, no entanto, depende da capacidade dos modelos de gerar sistemas realistas, ao menos do ponto de vista dos algoritmos de recuperação de arquitetura. Neste artigo propomos um modelo de síntese de *software* e avaliamos o realismo desse modelo e de um outro modelo presente na literatura.

Na próxima seção apresentamos os dois modelos avaliados. Na seção 4 descrevemos um experimento para avaliar os modelos quanto à capacidade de gerar sistemas semelhantes a sistemas reais. 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 recu-

perar 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, por essa razão, 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:

- 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 cujas medidas refletem os parâmetros fornecidos através de um procedimento iterativo. 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_1, então 40% das suas arestas estão ligadas a vértices que não pertencem ao módulo M_1.

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, enquanto em sistemas de *software* é comum controlar as dependências para facilitar a manutenção. Em sistemas estruturados em camadas, por exemplo, cada módulo depende de no máximo um outro módulo e as dependências não podem formar ciclos.

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+, uma extensão do modelo BCR que gera redes orientadas organizadas 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;

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

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

- 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.

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.

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 Distância entre Redes

Para avaliar o quanto uma rede sintética se parece com uma rede extraída de um sistema real, é conveniente considerar uma métrica de distância entre duas redes. Neste trabalho consideramos uma métrica baseada no conceito de matriz de vizinhança [1]. Essa métrica se aplica apenas a redes com o mesmo número de vértices.

A matriz de vizinhança de uma rede, M, é uma matriz $|V| \times |V|$ na qual cada elemento M_{ij} representa o comprimento do caminho mínimo entre dois vértices i e j. No caso de pares de vértices que não são conectados por um caminho, considera-se que o comprimento é 0. A matriz de vizinhança de uma rede não é única: sua forma exata depende da ordem em que os vértices aparecem na matriz.

A distância entre duas matrizes de vizinhança, $\mathrm{d}(M, N)$, é dada pela equação a seguir:

$$\mathrm{d}(M,N) = \sqrt{\frac{1}{|V|^2}\sum_{i,j=1}^{|V|}(M_{ij} - N_{ij})^2}$$

Sejam A e B duas redes com $|V|$ vértices cada uma, A_0 uma matriz de vizinhança qualquer de A e B^* o conjunto de todas as matrizes de vizinhança da rede

B — uma matriz para cada uma das possíveis $|V|!$ ordenações dos vértices de B. Definimos a distância entre as duas redes, $\mathrm{D}(A, B)$, como sendo a menor das distâncias entre a matriz A_0 e as matrizes do conjunto B^*:

$$\mathrm{D}(A, B) \;=\; \min\{d(A_0, x) \mid x \in B^*\}$$

Dada a necessidade de se considerar $|V|!$ casos, o cálculo exato da distância mínima é viável apenas para redes muito pequenas. Felizmente é possível obter um limite superior para essa distância através de uma busca heurística no conjunto B^*. Neste trabalho calculamos uma aproximação para $\mathrm{D}(A, B)$ através do algoritmo Monte Carlo descrito em [1].

4 Avaliação Empírica

Na seção 2 discutimos características gerais dos modelos de geração de redes. Nesta seção descrevemos um experimento realizado com o propósito de responder à seguinte pergunta: os modelos são capazes de produzir redes sintéticas semelhantes a redes extraídas de sistemas reais? Consideramos que um modelo é bem sucedido se ele é capaz de sintetizar uma rede cuja distância para uma rede real tomada como referência não seja maior do que a distância entre redes reais com características semelhantes.

Neste experimento consideramos um sistema de referência, o jogo VilloNanny 2.2.4, e um outro sistema com aproximadamente o mesmo número de classes, a biblioteca JFreeChart 1.0.13. Extraímos as redes de ambos os sistemas e calculamos algumas métricas a partir das redes. As métricas do VilloNanny foram usadas para ajustar os parâmetros dos modelos. A seguir usamos sintetizamos diversas redes e então calculamos a distância entre a rede do VilloNanny e as demais redes – a rede do JFreeChart e as redes sintéticas. Ignoramos o sentido das arestas de todas as redes, uma vez que o modelo LFR não produz redes orientadas.

4.1 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.

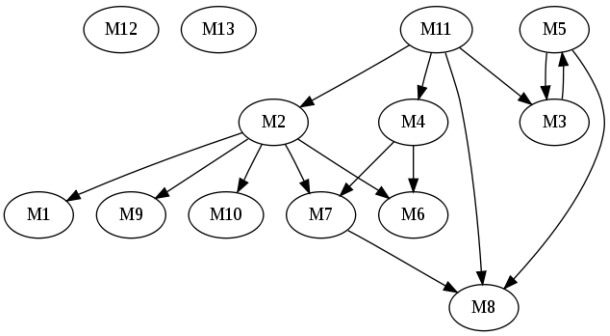

Figura 1: Arquitetura modular extraída do sistema VilloNanny.

As dependências entre as classes foram extraídas com a ferramenta DepFind[3] e então armazenadas como redes não-orientadas. A seguir removemos seis vértices com grau zero da rede do sistema JFreeChart, a fim de igualar o número de vértices das duas redes. Essa etapa foi necessária porque a métrica de distância entre redes só pode ser aplicada a duas redes que possuem o mesmo tamanho.

4.2 Caracterização das Redes Reais

A partir das redes extraídas na etapa anterior, calculamos diversas métricas, apresentadas na Tabela 4.2. Os expoentes das distribuições de graus e de tamanhos dos módulos foram estimados através do método da máxima verossimilhança[4] [5].

Além disso, extraímos a arquitetura modular do VilloNanny (Figura 4.2). Na arquitetura, existe uma aresta do módulo M_i para um módulo M_j somente se pelo menos um vértice de M_i se liga a um vértice de M_j.

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

[4] Utilizamos uma implementação disponível em `http://www.santafe.edu/~aaronc/powerlaws/`

Tabela 1: Métricas extraídas dos sistemas analisados: número de vértices, $|V|$; número de arestas, $|E|$; número de arestas externas $|E_{ext}|$; grau médio, $\langle k \rangle$; grau máximo, k_{max}; número de módulos, $|M|$; tamanho do menor módulo, t_{min}; tamanho do maior módulo, t_{max}; expoente da distribuição de graus, γ; expoente da distribuição dos tamanhos dos módulos, β.

Sistema	$\|V\|$	$\|E\|$	$\|E_{ext}\|$	$\langle k \rangle$	k_{max}	$\|M\|$	t_{min}	t_{max}	γ	β
VilloNanny	1658	6766	343	6,92	80	13	6	446	2,68	1,00
JFreeChart	1658	8296	1161	10,29	145	8	6	609	2,68	0,96

Tabela 2: Discrepâncias relativas entre as medidas das redes sintéticas (média) e as medidas da rede de referência.

	$\|E\|$	$\|E_{ext}\|$	$\|M\|$	γ	β
LFR	0,44%	0,06%	27,35%	4,10%	4,44%
BCR+	2,45%	2,17%	0,00%	19,69%	1,67%

Tabela 3: Distâncias das redes sintéticas e da rede do sistema JFreeChart em relação à rede de referência.

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

4.3 Escolha de Parâmetros e Síntese de Redes

Queremos gerar redes com medidas próximas às medidas da rede de referência, e isso depende de uma escolha adequada dos parâmetros dos modelos.

No modelo LFR os valores dos parâmetros $|V|$, $\langle k \rangle$, k_{max}, γ, β, t_{min} e t_{max} coincidem com os valores das métricas correspondentes extraídas da rede de referência. O parâmetro de mistura, μ, foi calculado segundo a equação $\mu = |E_{ext}| \ / \ |E|^s$. A seguir são listados os valores escolhidos para os parâmetros do modelo LFR: $|V| = 1658$; $\langle k \rangle = 6,92$; $k_{max} = 80$; $\gamma = 2,68$; $\beta = 1,00$ $\mu = 0,54$; $t_{min} = 6$; $t_{max} = 446$.

Para o modelo BCR+ utilizamos a arquitetura modular da Figura 4.2, extraída da rede de referência. Como estamos ignorando o sentido das arestas, consideramos $p = q$ e $\delta_{in} = \delta_{out}$. Feita essa restrição, ajustamos os parâmetros por tentativa e erro e chegamos aos seguintes valores: $|V| = 1658$; $p = q = 0,1$; $r = 0,8$; $\mu = 0,09$; $\delta_{in} = \delta_{out} = 3$.

Com os parâmetros escolhidos, geramos 9 redes a partir de cada modelo e extraímos métricas. Calculamos a média das medidas das redes de cada modelo e então as comparamos com as medidas da rede de referência. A Tabela 4.3 mostra, para cada modelo, a discrepância entre as medidas da rede de referência e as médias das redes sintéticas do modelo.

4.4 Cálculo da Distância Entre as Redes

Após a extração das redes reais e da produção de redes sintéticas, calculamos as distâncias entre a rede de referência e todas as demais redes. Os resultados, agregados por modelo, podem ser vistos na Tabela 4.4. Os dados mostram que o modelo BCR+ foi o que mais se aproximou da rede de referência; uma das redes geradas ficou mais próxima à rede de referência do que o sistema JFreeChart. O modelo LFR produziu redes próximas umas às outras, como revela o pequeno desvio-padrão, porém mais distantes da rede de referência.

4.5 Ameaças à Validade

Os resultados são bastante animadores, mas precisam ser analisados à luz de algumas limitações do experimento:

- apenas dois sistemas reais foram estudados, e por isso, os resultados não podem ser generalizados;
- a métrica de distância ignora a organização das redes em módulos; assim, nada se pode afirmar sobre o realismo das associações entre vértices e módulos;

- muitos algoritmos de recuperação de arquitetura operam sobre redes orientadas e ponderadas, que contêm mais informação do que as redes não-orientadas que analisamos.

5 Discussão

Este artigo apresentou uma nova abordagem, baseada na geração de sistemas de *software* sintéticos, de avaliação de algoritmos de recuperação de arquitetura. Dois modelos de sistemas sintéticos foram apresentados e avaliados empiricamente. Um experimento mostrou que o modelo BCR+, proposto neste artigo, é capaz de sintetizar sistemas semelhantes a sistemas reais.

Dado que todo modelo é uma simplificação, os sistemas sintéticos não devem ser vistos como substitutos de sistemas reais, e sim como uma alternativa que apresenta diversas vantagens. Além de proporcionarem grandes volumes de dados para teste, eles permitem que se estude como a acurácia dos algoritmos é afetada por variações nos parâmetros dos modelos. Por exemplo, Lancichinetti, Fortunato e Radicchi estudaram dois algoritmos de aglomeração e mostraram que eles apresentam menor acurácia quando aplicados a redes com baixo grau médio [7].

Acreditamos que a síntese de *software* pode beneficiar pesquisas sobre outras tarefas de engenharia reversa, tais como análise de impacto e localização de funcionalidades. A avaliação de ferramentas de engenharia reversa em geral depende da disponibilidade de informações sobre sistemas que raramente são documentadas. No caso de localização de funcionalidades, por exemplo, essa informação é o mapeamento entre as funcionalidades de um sistema e trechos de seu código-fonte.

Dois modelos recentes de redes deverão ser avaliados em um trabalho futuro. O modelo LR é uma extensão do modelo LFR para grafos orientados e ponderados [6]. O modelo CGW produz redes orientadas através de um processo incremental que inclui remoção e religamento de arestas [4].

O próximo passo da pesquisa é aplicar algoritmos de recuperação de arquitetura às redes sintéticas. Poderemos, então, comparar os resultados com dados de experimentos já realizados com sistemas reais.

Referências

[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.