Redes Complexas e Dependências entre Componentes de *Software*

Rodrigo Rocha Gomes e Souza

10 de maio de 2009

Este relatório aborda redes de dependências entre componentes de *software*, explica conceitos da teoria das redes complexas e enumera trabalhos recentes que aplicam essa teoria à análise de redes de dependências entre componentes.

1 Redes de Dependências entre Componentes

Sistemas de *software* precisam ser modificados constamente. Dependências excessivas entre os componentes de um sistema podem tornar as modificações mais custosas, uma vez que dificultam a compreensão dos componentes isoladamente. Por essa razão o estudo de redes formadas por dependências entre componentes de *software* pode fornecer pistas sobre a atividade de desenvolvimento de *software* em geral.

Neste trabalho consideramos como componentes as entidades de código-fonte, tais como classes, métodos e atributos (no caso de linguagens orientadas objetos), ou procedimentos, funções e tipos abstratos de dados (no caso de linguagens procedimentais). Dizemos que um componente depende de outro quando o funcionamento do primeiro está condicionado à presença do segundo. Dependências podem se originar de diversos tipos de interação, como uma chamada a uma função ou a leitura de um atributo. Dependências são essencialmente assimétricas: o fato de um componente A depender de um componente B não implica que B depende de A.

1.1 Extração

A rede de dependências entre componentes de um sistema pode ser extraída automaticamente por um programa construído para este fim, denominado extrator de dependências. As principais abordagens de extração de dependências são a análise dinâmica e a análise estática.

A análise dinâmica envolve a instrumentação do sistema para coletar dados de sua execução, e por isso exige que o programa sob análise esteja corretamente instalado e configurado. Para que a análise seja efetiva, é preciso que o programa seja executado com as mais diversas entradas a fim de capturar

todos os comportamentos interessantes, o que normalmente é feito através da execução de casos de teste. Essa imposição restringe o conjunto de sistemas que podem ser efetivamente analisados dinamicamente, e nos demais sistemas o tempo de execução da análise pode ser proibitivo.

A análise estática é realizada sobre o código-fonte ou o código objeto do sistema, dispensando sua execução, mas tende a ser menos precisa do que a análise dinâmica e não detecta interações resultantes de introspecção. Apesar disso a análise estática é mais aplicável no caso geral, pois não depende da existência de testes automáticos representativos e é, em geral, mais rápida que a dinâmica.

Nenhuma técnica automática é capaz de determinar corretamente, em todos os casos, se existe uma dependência entre dois componentes — basta considerar o uso de ponteiros em linguagens como C++, ou a transferências de dados através de arquivos. Assim, redes de dependências extraídas automaticamente devem ser consideradas apenas aproximações das redes de dependências reais.

1.2 Simplificação e Detalhamento

A depender do tipo de análise que se deseja realizar sobre uma rede de dependências, pode ser conveniente filtrar ou contrair componentes, ou mesmo tratar dependências como relacionamentos simétricos. A filtragem consiste em remover componentes ou dependências consideradas desnecessárias para a análise. A contração pode ser feita quando há uma relação de composição entre os componentes, como no caso de classes (que são compostas de atributos e métodos). Nesse caso uma classe e todos os atributos e métodos que a compõem são contraídos, isto é, representados como um só componente, a classe. Desse processo surgem dependências implícitas: se na rede original há uma dependência entre dois métodos, na rede contraída aparece uma dependência implícita entre as classes que os contêm.

Há análises que, por outro lado, requerem informações mais detalhadas. Pode ser necessário considerar não apenas a existência de dependências, mas também identificar quais são os tipos de interação que ocorrem entre dois componentes e com que frequência essas interações ocorrem.

2 Redes Complexas

A teoria das redes complexas estuda propriedades gerais de diversos tipos de redes com o uso de ferramentas estatísticas. Estudos realizados na última década revelaram similaridades entre diversas redes, sejam elas tecnológicas, como a Web e a rede de distribuição de energia elétrica dos Estados Unidos, biológicas, como cadeias alimentares e ligações entre proteínas, ou sociais, como as relações de amizade entre alunos de uma escola. Essas redes têm sido caracterizadas por observações como a distribuição de graus, o efeito mundo pequeno, o coeficiente de agrupamento e a presença de motivos.

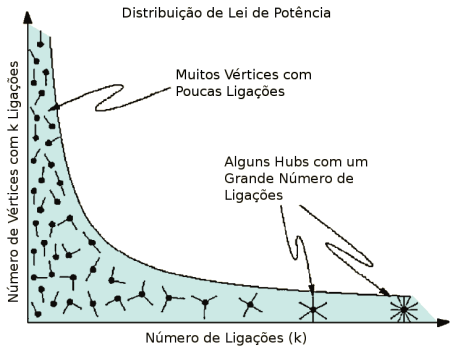

Figura 1: Distribuição de graus como lei de potência. Adaptado de [Bar07].

2.1 Distribuição de Graus

Barabási e Albert [BA99] analisaram uma amostra da *World Wide Web*, modelada como um grafo não-orientado no qual os vértices representam páginas e as arestas representam *links* entre duas páginas. Eles perceberam que a probabilidade de um vértice escolhido ao acaso ter grau k (isto é, estar ligado a k arestas) seguia uma lei de potência, $p(k) \sim k^{-\gamma}$, como mostra a Figura 2.1. Dizemos, nesse caso, que a rede possui uma distribuição de graus livre de escala, ou simplemente que a rede é livre de escala. Desde então esse tipo de distribuição foi encontrado em diversas redes, incluindo redes de dependências entre componentes de programas de computador [VS03].

Esse resultado foi considerado surpreendente, pois contradiz a hipótese de que redes reais obedecem ao modelo de Erdős-Rényi [ER59], também chamado de modelo de rede aleatória. Nesse modelo, a probabilidade de um par de vértices ser ligado por uma aresta é constante e igual a p. Demonstra-se que a distribuição dos graus de redes geradas por esse modelo é bem aproximada pela distribuição de Poisson.

Uma característica das redes livres de escala é a presença de vértices cujo grau é muito maior do que a média (informalmente chamados de *hubs*). No caso de redes de Erdős-Rényi a probabilidade de existir um vértice com um determinado grau k cai exponencialmente à medida que k se afasta do valor médio e, por essa razão, nessas redes os vértices possuem mais ou menos o mesmo grau e a existência de *hubs* é altamente improvável.

Barabási e Albert propuseram um modelo para explicar a formação das redes com distribuição de graus livre de escala, formado por dois mecanismos: crescimento contínuo e ligação preferencial. O modelo propõe que as redes crescem um vértice por vez e cada novo vértice se liga a um número fixo de vértices antigos, dando preferência aos vértices com maior grau (mais formalmente, a probabilidade de um vértice receber uma aresta é proporcional ao seu grau). Hoje sabe-se que redes livres de escala também podem ser geradas por outros modelos [AB00, KRR$^+$00, ACL00, DGM02, BBCR03, DC05].

2.1.1 Efeito Mundo Pequeno

A distância entre dois vértices de uma rede é o número de arestas do menor caminho que conecta os vértices. Diz-se que uma rede apresenta o efeito mundo pequeno quando a distância entre dois vértices é, em média, pequena, mesmo quando a rede é grande. Mais formalmente, a distância média entre vértices é proporcional ao logaritmo do número de vértices [WS98]. Esse efeito foi detectado em diversas redes reais.

2.1.2 Coeficiente de agrupamento

Os vizinhos de um vértice são todos os vértices com os quais ele compartilha uma aresta. O coeficiente de agrupamento de um vértice, C_i, é a fração de todos os possíveis pares de vizinhos do vértice que estão ligados por uma aresta, e é dado pela seguinte expressão:

$$C_i = \frac{2x}{k_i(k_i - 1)}$$

onde x é o número de pares de vizinhos do vértice i que estão ligados por uma aresta e k_i é o grau do vértice i [WS98]. Por definição, $C_i = 0$ quando $k_i < 2$. Define-se o coeficiente de agrupamento de um grafo, C, como a média aritmética dos coeficientes de agrupamento dos seus vértices. Demonstra-se que o coeficiente de agrupamento de uma rede aleatória de Erdős-Rényi é igual a $\langle k \rangle / n$ (onde $\langle k \rangle$ é o grau médio). Muitas redes complexas possuem um coeficiente de agrupamento alto, isto é, muito maior do que o coeficiente das redes aleatórias.

Outra característica observada em algumas redes é que o coeficiente de agrupamento de um vértice é inversamente proporcional ao grau do vértice, ou seja, $C(k) \sim k^{-1}$. Segundo Ravasz e Barabási [RB03], isso indica a existência de uma organização hierárquica na rede.

2.1.3 Motivos

Motivos são padrões de vértices e arestas que ocorrem com frequência em uma rede [MSOI$^+$02]. Estudos recentes encontraram em redes de dependências entre componentes motivos presentes em redes de neurônios e em circuitos eletrônicos [VS05, MHL08]. Especula-se que a formação de motivos em sistemas de *software* seja resultado de restrições de custo e otimização impostas a sua evolução.

2.1.4 Detecção de Comunidades

Métricas globais como a distribuição de graus escondem o fato de que muitas redes se organizam em grupos de vértices relativamente independentes uns dos outros, chamados de comunidades. Detecção de comunidades é a tarefa de particionar os vértices de uma rede em comunidades de modo que, sob algum critério, os vértices de uma mesma comunidade estão fortemente relacionados entre si e fracamente relacionados a vértices de outras comunidades.

A detecção de comunidades é análoga ao que, na mineração de dados, se chama de análise de clustering — sendo cluster, nesse contexto, sinônimo de comunidade. No domínio da engenharia de sofware, muitas técnicas de recuperação de arquitetura utilizam algoritmos de clustering para identificar módulos de um sistema de software.

A avaliação de algoritmos de clustering (ou detecção de comunidades) consiste em executar o algoritmo sobre uma rede cuja estrutura de clusters (ou comunidades) é conhecida e então comparar, através de alguma métrica, essa estrutura com a estrutura encontrada pelo algoritmo. Infelizmente não há muitas redes cuja estrutura de comunidades é conhecida, e as que existem são pequenas. Testes mais extensivos podem ser realizados com redes geradas por computador.

Newman e Girvan [NG04] propuseram um modelo que gera redes com 128 vértices, divididos em quatro comunidades de 32 vértices cada. Os vértices possuem aproximadamente o mesmo número de arestas, 16, das quais z_{out} arestas ligam vértices de comunidades distintas. Newman e Girvan geraram redes com diferentes valores de z_{out} com a finalidade de comparar dois algoritmos de detecção de comunidade.

Esse modelo possui algumas limitações: os vértices possuem aproximadamente o mesmo número de arestas, as comunidades possuem o mesmo tamanho e a rede é pequena. Com base nessa observação, Lancichinetti et al. [LFR08] propuseram um modelo no qual a distribuição de graus e a distribuição de tamanho das comunidades são ambas leis de potência e o número de vértices é arbitrário. Cada vértice compartilha uma fração constante de suas arestas com vértices de outras comunidades; essa fração é controlada pelo parâmetro de mistura, μ.

O modelo de Lancichinetti et al. possui características que, a depender do domínio modelado, são pouco realistas: as redes são não-orientadas; todos os vértices possuem ligações tanto para vértices da mesma comunidade quanto para vértices de outras comunidades; todas as comunidades estão, potencialmente, ligadas a todas as outras.

3 Redes de Dependências entre Componentes como Redes Complexas

Estudos recentes têm aplicado a teoria das redes complexas em redes de dependências entre componentes de *software*. Valverde e Solé [VS03] detectaram distribuições de graus livres de escala e alto coeficiente de agrupamento em redes não-orientadas formadas por relações de agregação de tipos em diagramas UML, programas em C e programas em C++. Myers [Mye03] analisou redes de chamadas de função em programas em C e redes de agregação e herança em programas em C++, modeladas como grafos orientados. Em ambos os casos ele identificou organização hierárquica através da distribuição do coeficiente de agrupamento, $C(k) \sim k^{-1}$. O estudo revelou ainda uma correlação negativa entre o grau de entrada e o grau de saída de um vértice. Leis de potência foram

encontradas em trechos das distribuições de graus de entrada e das distribuições de graus de saída.

Distribuições de graus livres de escala também foram encontradas em programas escritos em Smalltalk [MPST04, CMPS07] e em Java [HWCK06, BFN$^+$06, IMI08], em dependências entre pacotes de *software* [LW04], em chamadas de sistema, em dependências entre bibliotecas dinâmicas [LSV08] e até mesmo em referências entre objetos em tempo de execução [PNFB05].

É difícil comparar as pesquisas porque nem sempre elas deixam explícito que tipos de interação foram considerados para a construção das redes. Além disso, alguns trabalhos usam ferramentas estatísticas inadequadas para leis de potência.

4 Tratamento Estatístico de Leis de Potência

4.1 Histograma

A estratégia mais usada para determinar se um conjunto de dados segue uma lei de potência é plotar o seu histograma em escala log-log. Se os dados seguem a lei de potência, o gráfico resultante deve ser uma reta. No entanto alguns cuidados precisam ser tomados ao traçar o histograma.

Quando o histograma agrupa os dados em classes de mesmo tamanho, o gráfico apresenta muito ruído na cauda, na região onde se encontram os maiores valores, como mostra a Figura 4.1(b). Isso ocorre porque nessa região os dados são esparsos, já que a probabilidade é baixa.

Uma maneira de reduzir o ruído consiste em agrupar os dados em classes cujo tamanho cresce exponencialmente. Por exemplo, podemos considerar classes formadas pelos intervalos de 1 a 2, de 2 a 4, de 4 a 8, de 8 a 16, e assim sucessivamente. A contagem de amostras de uma classe, nesse caso, deve ser dividida pelo tamanho do intervalo, a fim de normalizar o histograma [New05]. Note como o ruído foi reduzido na Figura 4.1(c).

Uma solução um pouco diferente é traçar o gráfico da função de distribuição cumulativa complementar (fdcc). Essa função representa, para cada valor x, o número de elementos com valor maior do que x. Não há necessidade de dividir os dados em classes. Se os dados seguem uma lei de potência com expoente γ, sua fdcc será uma lei de potência com expoente γ–1. O gráfico é mostrado na Figura 4.1(d).

4.2 Estimação de Parâmetros

Poucos dados empíricos seguem uma lei de potência em toda a sua extensão. Normalmente a lei de potência se aplica apenas a valores maiores do que um valor mínimo, x_{min}, e nesse caso se diz que a cauda da distribuição segue uma lei de potência. O parâmetro x_{min} pode ser estimado visualmente através da observação do histograma ou através de métodos estatísticos [CSN07].

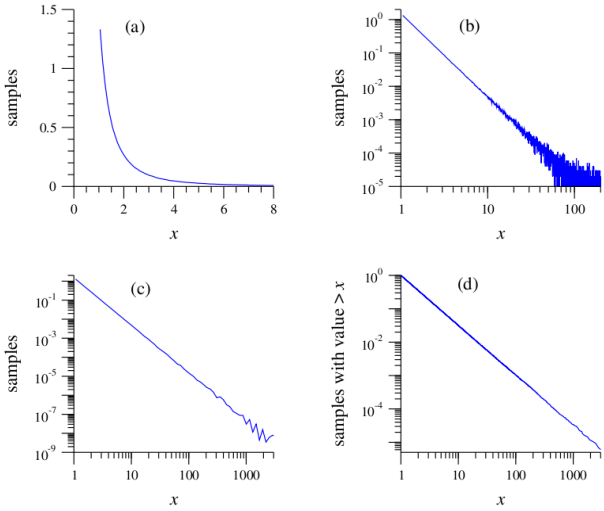

Figura 2: (a) Histograma de um conjunto de 1 milhão de números aleatórios, com distribuição de lei de potência com $\gamma = 2,5$. (b) O mesmo histograma, em escala log-log. (c) Histograma construído com classes de tamanho exponencial. (d) A função de distribuição cumulativa complementar do mesmo conjunto de dados. Fonte: [New05].

Ao extrair o logaritmo da lei de potência, obtemos $\log(p(k)) \sim -\gamma \log(k)$, que é uma reta com coeficiente angular $-\gamma$. Por essa razão uma técnica comum para estimar o parâmetro γ de um conjunto de dados consiste em aplicar o método dos mínimos quadrados sobre o logaritmo dos dados a fim de encontrar o coeficiente angular da reta. Esse método, no entanto, introduz erros sistemáticos no caso da lei de potência, e por isso recomenda-se o uso do método da máxima verossimilhança [CSN07]. O

4.3 Teste de Ajustamento

Estimados os parâmetros x_{min} e γ, convém verificar se os dados se ajustam bem a uma lei de potência. Um valor alto para o coeficiente de determinação, R^2, não é suficiente para afirmar que o ajuste é bom. Uma forma melhor de realizar a verificação consiste em gerar leis de potência com os parâmetros estimados e compará-las aos dados através do teste de ajustamento de Kolmogorov-Smirnov. Para um maior rigor, é preciso ajustar outras distribuições teóricas aos dados e verificar qual se ajusta melhor.

A depender da aplicação pode não ser importante determinar se a distribuição dos dados segue perfeitamente uma lei de potência. Em muitos casos é suficiente observar que os dados são distribuídos de forma bastante heterogênea, formando uma cauda pesada, que decai lentamente.

Referências

[AB00] Reka Albert and Albert-Laszlo Barabasi. Topology of evolving networks: local events and universality. *Physical Review Letters*, 85:5234, 2000.

[ACL00] William Aiello, Fan Chung, and Linyuan Lu. A random graph model for power law graphs. *Experimental Math*, 10:53–66, 2000.

[BA99] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

[Bar07] Albert-Laszlo Barabasi. The architecture of complexity: From network structure to human dynamics. *Control Systems Magazine, IEEE*, 27:33–42, August 2007.

[BBCR03] Béla Bollobás, Christian Borgs, Jennifer Chayes, and Oliver 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.

[BFN+06] Gareth Baxter, Marcus Frean, James Noble, Mark Rickerby, Hayden Smith, Matt Visser, Hayden Melton, and Ewan Tempero. Understanding the shape of java software. *SIGPLAN Not.*, 41(10):397–412, 2006.

[CMPS07] Giulio Concas, Michele Marchesi, Sandro Pinna, and Nicola Serra. Power-laws in a large object-oriented software system. 33(10):687–708, 2007.

[CSN07] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data, 2007.

[DC05] Narsingh Deo and Aurel Cami. A birth-death dynamic model of scale-free networks. In *ACM-SE 43: Proceedings of the 43rd annual Southeast regional conference*, pages 26–27, New York, NY, USA, 2005. ACM.

[DGM02] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. Pseudofractal scale-free web. *Physical Review E*, 65:066122, 2002. Citado por Myers, Barabási.

[ER59] P. Erdös and A. Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[HWCK06] David Hyland-Wood, David Carrington, and Simon Kaplan. Scale-free nature of java software package, class and method collaboration graphs. In *Proceedings of the 5th International Symposium on Empirical Software Engineering, Rio de Janeiro, Brasil*, 2006.

[IMI08] M. Ichii, M. Matsushita, and K. Inoue. An exploration of power-law in use-relation of java software systems. In *Proc. 19th Australian Conference on Software Engineering ASWEC 2008*, pages 422–431, 2008.

[KRR+00] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 57, Washington, DC, USA, 2000. IEEE Computer Society. Copying model.

[LFR08] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 78(4), 2008.

[LSV08] Panagiotis Louridas, Diomidis Spinellis, and Vasileios Vlachos. Power laws in software. *ACM Trans. Softw. Eng. Methodol.*, 18(1):1–26, 2008.

[LW04] Nathan LaBelle and Eugene Wallingford. Inter-package dependency networks in open-source software, 2004.

[MHL08] Yutao Ma, Keqing He, and Jing Liu. Network motifs in object-oriented software systems. *CoRR*, abs/0808.3292, 2008.

[MPST04] Michele Marchesi, Sandro Pinna, Nicola Serra, and Stefano Tuveri. Power laws in smalltalk. In *ESUG Conference*, Kothen, Germany, September 2004.

[MSOI+02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, October 2002.

[Mye03] Christopher 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.

[New05] M. E. J. Newman. *Power laws, Pareto distributions and Zipf's law*, 2005.

[NG04] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[PNFB05] Alex Potanin, James Noble, Marcus Frean, and Robert Biddle. Scale-free geometry in oo programs. *Commun. ACM*, 48(5):99–103, 2005.

[RB03] Erzsebet Ravasz and Albert-Laszlo Barabasi. Hierarchical organization in complex networks. *Physical Review E*, 67:026112, 2003.

[VS03] Sergi Valverde and Ricard V. Solé. Hierarchical small worlds in software architecture. (Directed Scale-Free Graphs), 2003.

[VS05] Sergi Valverde and Ricard V. Solé. Network motifs in computational graphs: A case study in software architecture. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 72(2), 2005.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.