Modelos Estocásticos para Síntese de Sistemas de Software Organizados em Módulos *DRAFT v3*

Rodrigo Rocha Gomes e Souza

24 de maio de 2009

Resumo

Algoritmos de modularização de *software* procuram identificar a estrutura de módulos de um sistema de *software* a partir de sua implementação. Infelizmente, dada a dificuldade de se encontrar sistemas cuja organização modular é bem documentada, há poucos estudos dedicados a testar empiricamente esses algoritmos. Propomos suprir essa escassez através do uso de modelos estocásticos capazes de gerar sistemas de *software* organizados em módulos para servirem como conjunto de teste. Neste artigo apresentamos dois modelos que cumprem essa função, um deles inédito, e concluímos através de um experimento que um dos modelos produz sistemas semelhantes a sistemas reais, ao menos do ponto de vista dos algoritmos de modularização.

1 Introdução

A divisão conceitual de um sistema de *software* em módulos é uma informação valiosa durante o seu desenvolvimento, sobretudo para a divisão de tarefas entre desenvolvedores [6]. Essa informação muitas vezes não é documentada adequadamente e, à medida que os desenvolvedores de um sistema são substituídos, pode se perder. Por essa razão foram propostos diversos algoritmos de modularização de *software*, os quais procuram identificar a organização em módulos de um sistema de *software* a partir de sua implementação [9, 2, 10].

Espera-se que os algoritmos encontrem organizações modulares semelhantes àquelas que seriam encontradas por especialistas nos sistemas analisados. Uma forma de testar os algoritmos consiste, pois, em aplicá-los ao código-fonte de um sistema cuja organização em módulos esteja documentada e então comparar as duas estruturas de módulos [12]. Infelizmente é difícil encontrar sistemas documentados nesse nível de detalhe e por essa razão a avaliação empírica dos algoritmos de modularização de *software* ainda é incipiente.

Para suprir a escassez de sistemas bem documentados, propomos o uso de modelos estocásticos capazes de sintetizar sistemas de *software* organizados em módulos para servirem como conjunto de teste. Para que essa abordagem seja bem-sucedida, no entanto, é preciso que os sistemas sintetizados pelos modelos sejam semelhantes a sistemas reais, ao menos do ponto de vista dos algoritmos de modularização. Neste artigo apresentamos dois modelos, um deles inédito, e avaliamos através de um experimento o grau de semelhança entre sistemas reais e sistemas sintetizados pelos modelos.

A próxima seção apresenta o embasamento teórico deste trabalho. A seguir, na seção 3, descrevemos os modelos estocásticos que sintetizam sistemas de software organizados em módulos. Na seção 4 relatamos o experimento realizado para avaliar os modelos e os seus resultados. Por fim, discutimos direções futuras para pesquisa.

2 Teoria

Sistemas de software, no nível de implementação, são formados por vários componentes que interagem entre si de diversas formas. Sistemas implementados em linguagens orientadas a objetos, por exemplo, são compostos por classes que interagem com outras classes através de mecanismos como troca de mensagens e herança. As interações es-

tabelecem relações de dependência entre os componentes, de forma que cada componente só funciona corretamente na presença dos componentes dos quais ele depende.

Para viabilizar o desenvolvimento de um sistema de *software* com muitos componentes é comum agrupar os componentes em módulos. Não existe um critério unânime para realizar esse agrupamento, mas em geral se assume que os módulos agrupam componentes que dependem mais uns dos outros do que de componentes que pertencem a outros módulos. Esse padrão de dependências permite que os módulos sejam desenvolvidos de forma mais ou menos independente, possivelmente por equipes de desenvolvimento distintas.

A rede de dependências entre os componentes de um sistema pode ser extraída automaticamente através de ferramentas de análise estática de código-fonte ou código objeto. A organização dos componentes em módulos, por outro lado, não é explicitamente representada no código-fonte. O objetivo das ferramentas de modularização de software é encontrar, a partir da rede de dependências de um sistema, uma organização dos componentes em módulos que seja adequada para apoiar o desenvolvimento do sistema.

A rede de dependências entre componentes de um sistema de *software* pode ser representada por um grafo no qual os vértices representam os componentes e as arestas representam as relações de dependência entre os componentes. Neste trabalho usamos os termos rede e grafo como sinônimos. A depender do nível de detalhe desejado, o grafo pode ser não-orientado — no qual considera-se que a relação de dependência é simétrica — ou orientado — no qual as arestas possuem uma origem e um destino bem definidos. Pode-se ainda associar a cada aresta um número indicando o peso da dependência, o que caracteriza o grafo como ponderado. A organização em módulos é representada através de um particionamento do conjunto de vértices, no qual cada partição representa um módulo. A Figura 1 mostra um grafo orientado, não-ponderado e organizado em módulos.

A arquitetura modular de um sistema organizado em módulos é um grafo que representa os módulos e as dependências entre eles. Na arquitetura modular há uma aresta de um módulo M_1 para um módulo M_2 somente se existe uma aresta de um vértice de M_1 para um vértice de M_2. A Figura 2 mostra a arquitetura modular da Figura 1.

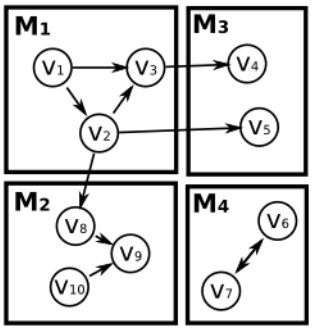

Figura 1: Representação gráfica de uma rede orientada organizada em quatro módulos.

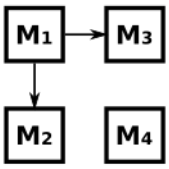

Figura 2: A arquitetura modular da rede da Figura 1.

Pesquisas recentes mostram que redes de dependências entre componentes possuem características comuns a redes complexas estudadas em diversos domínios, tais como sociologia, biologia e linguística [11, 13]. Uma das características mais estudadas é a distribuição dos graus dos vértices. Descobriu-se que, em diversas redes estudadas, a distribuição dos graus segue uma lei de potência, ou seja, o número de vértices ligados a exatamente k arestas é proporcional a $k^{-\gamma}$, onde o expoente γ varia tipicamente entre 2 e 3. Uma consequência dessa distribuição é a presença de vértices cujo grau é muito superior à média.

As características comuns a redes de diversos domínios foram incorporadas a vários modelos estocásticos de geração de redes [3, 8]. As redes geradas por esses modelos podem ser interpretadas como redes de dependências entre componentes.

2.1 Distância entre Redes

Embora métricas como o expoente da distribuição de graus (γ) sejam úteis para caracterizar redes, elas não são adequadas para se diferenciar duas redes. Não é difícil encontrar redes que, embora possuam o mesmo número de vértices, o mesmo número de arestas e o mesmo expoente γ, são muito diferentes entre si. Por essa razão neste trabalho usamos uma métrica de distância entre redes [1] para avaliar o quanto uma rede sintética se parece com uma rede extraída de um sistema real. Essa métrica se baseia no conceito de matrizes de vizinhança e se aplica apenas a redes com o mesmo número de vértices.

Uma 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 do i-ésimo até o j-ésimo vértice. No caso de pares de vértices que não são conectados por um caminho, considera-se que o comprimento é 0. De acordo com a definição, a construção da matriz de vizinhança de uma rede depende de uma permutação dos vértices da rede e, por isso, uma rede com $|V|$ vértices define $|V|!$ matrizes de vizinhança.

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. 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 $\mathrm{D}(A, B)$ obtida através de um algoritmo Monte Carlo descrito em [1].

3 Modelos Estocásticos

Esta seção apresenta dois modelos de geração de redes organizadas em módulos que podem ser usados para sintetizar redes de dependências entre componentes. Primeiramente descrevemos um modelo descrito na literatura sobre redes complexas e, a seguir, apresentamos um novo modelo.

3.1 O Modelo LFR

Lancichinetti, Fortunato e Radicchi [8] 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,

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

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.

3.2 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;
- 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.

[2] Uma implementação pode ser baixada em `http://code.google.com/p/swasr/wiki/Downloads`

4 Avaliação Empírica

Na seção 3 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 escolhemos o jogo VilloNanny 2.2.4 como sistema de referência. O sistema JFreechart 1.0.13 é aproximadamente do mesmo tamanho e por isso é usado como base de comparação. 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. Essa é uma aproximação razoável, uma vez que arquivos JAR distintos são, em geral, desenvolvidos por equipes diferentes.

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.

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

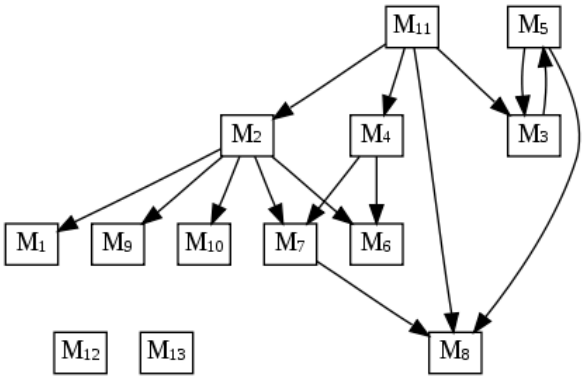

Figura 3: Arquitetura modular extraída do sistema VilloNanny.

4.2 Caracterização das Redes Reais

A partir das redes extraídas na etapa anterior, calculamos diversas métricas, apresentadas na Tabela 1. 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 3). 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.

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 $\mu = |E_{ext}| \ / \ |E|$. 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 3, extraída da rede de referên-

[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

cia. 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: $|V| = 1658$; $p = q = 0,1$; $r = 0,8$; $\mu = 0,09$; $\delta_{in} = \delta_{out} = 3$.

Escolhidos os parâmetros, geramos nove 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 2 mostra, para cada modelo, a discrepância entre as medidas da rede de referência e as médias das redes sintéticas do modelo. Na maioria dos casos a discrepância pode ser creditada ao tamanho das redes, relativamente pequeno — os modelos garantem que algumas medidas se aproximam da medida esperada quando o tamanho da rede tende a infinito. Em dois casos, no entanto, a discrepância foi maior do que 5%, o que sugere que a escolha dos parâmetros pode ser aprimorada.

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 3. 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 entre si, como revela o pequeno desvio-padrão, porém mais distantes da rede de referência.

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	$2,44 \pm 0,00$	$2,44$

4.5 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 modularização de *software* 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 modularização de *software*. Dois modelos de sistemas sintéticos foram apresentados e avaliados através de um experimento. O experimento mostrou que o modelo BCR+, proposto neste artigo, é capaz de sintetizar sistemas semelhantes a sistemas reais.

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%

Não defendemos que os sistemas sintéticos devem substituir completamente os sistemas reais no teste de algoritmos de modularização de *software*, mas acreditamos que eles podem proporcionar novas percepções sobre os algoritmos. Além de fornecerem um grande volume de dados para teste, eles permitem que se estude como a acurácia dos algoritmos é afetada por variações nos parâmetros dos modelos. Lancichinetti, Fortunato e Radicchi, por exemplo, estudaram dois algoritmos de modularização de redes e mostraram que eles apresentam resultados piores quando aplicados a redes com baixo grau médio [8].

Acreditamos que modelos de *software* pode beneficiar pesquisas em outras áreas 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 não são representadas explicitamente em seu código-fonte. No caso de localização de funcionalidades, por exemplo, essa informação é o mapeamento entre as estruturas da implementação de um sistema e as suas funcionalidades. Se esse mapeamento for incorporado em um dos modelos aqui apresentados, então será possível testar as ferramentas de localização de funcionalidades com um grande volume de dados.

Dois modelos de redes propostos recentemente, LR e CGW, deverão ser avaliados em um trabalho futuro. O modelo LR é uma extensão do modelo LFR para redes orientadas e ponderadas [7]. Já 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 modularização de *software* a redes sintéticas. Os resultados desse experimento poderão ser comparados com os resultados 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] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little. *Documenting Software Architectures: Views and Beyond.* Pearson Education, 2002.

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

[8] 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.

[9] 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.

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

[11] 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.

[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.