Artigo 1

Rodrigo Rocha Gomes e Souza

ABSTRACT

1. INTRODUÇÃO

A divisão conceitual de um sistema de *software* em módulos é muito importante no gerenciamento da evolução do sistema (Parnas, 1972). 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.

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-lo ao código-fonte de um sistema cuja organização em módulos seja conhecida e então comparar os módulos do sistema com os módulos encontrados pelo algoritmo através de uma medida de similaridade entre particionamentos (cite mojo, edgemojo etc.). Infelizmente é difícil encontrar sistemas documentados nesse nível de detalhe, e por essa razão a avaliação empírica dos algoritmos de aglomeração no domínio de sistemas de software ainda é insuficiente.

Propomos uma abordagem de avaliação complementar, baseada na produção 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 avaliar algoritmos de agrupamento com amostras arbitrariamente grandes de sistemas arbitrariamente complexos e, sobretudo, estudar como o desempenho dos algoritmos é afetado por parâmetros que definem os sistemas sintéticos.

Neste artigo apresentamos um modelo para a geração de sistemas sintéticos e avaliamos o realismo desse modelo, isto é, a similaridade entre os sistemas sintéticos e sistemas reais. Para fins de comparação, essa mesma análise é realizada sobre modelos genéricos disponíveis na literatura sobre redes complexas.

Na próxima seção.... na seção x.... por fim, ...

2. FUNDAMENTAÇÃO TEÓRICA

this work, we present an empirical study in which we evaluate clustering algorithms that work on design extracted from source code through static analysis.

Neste trabalho consideramos Embora a recuperação de módulos tenha sido aplicada a diversos paradigmas de programação, neste trabalho nos concentramos em sistemas orientados a objetos. A recuperação de módulos de um sistema é dividida em duas etapas: extração de dependências e aglomeração. Considerando

Neste trabalho consideramos apenas a recuperação de módulos a partir da implementação de sistemas orientados a objetos.

2.1 Redes de Dependências entre Classes

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.

2.2 Modelos de Geração de Redes Complexas

Pesquisas recentes mostram que redes de dependências entre classes possuem características comuns a redes complexas estudadas em diversos domínios, tais como sociologia, biologia e linguística [6, 7]. Foram desenvolvidos diversos modelos que simulam a construção de redes com tais características [2, 5]; esses modelos podem ser usados, portanto, para gerar redes de dependências entre classes.

2.2.1 Modelo Aleatório com Módulos

O modelo de Erdős-Rényi [4], ou modelo de rede aleatória, gera redes não-orientadas cujo número de arestas por vértice segue uma distribuição de Poisson. Ele não modela adequadamente as redes de software, e por isso é usada apenas como base de comparação. Propomos aqui uma simples extensão desse modelo que considera a organização dos vértices em módulos. O modelo possui os seguintes parâmetros:

- número de vértices, $|V|$;

- probabilidade de ligação, p;
- número de módulos, C.

O algoritmo de geração de redes inicia-se com a criação de $|V|$ vértices. A seguir cada vértice é incluído em um módulo escolhido com probabilidade uniforme. Por fim, para cada par de vértices, é adicionada uma aresta, com probabilidade p.

2.2.2 Modelo LFR

Lancichinetti, Fortunato e Radicchi [5] criaram um modelo de rede com estrutura de módulos embutida. O modelo não foi baseado em nenhum domínio em particular, mas incorpora distribuições estatísticas encontradas em redes de vários domínios. 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 x^{β}, com β entre 1 e 2.

O modelo possui os seguintes parâmetros:

- quantidade de vértices, N;
- expoente da distribuição de graus, γ_2;
- grau médio, $\langle k \rangle$;
- grau máximo, k_{max};
- expoente da distribuição de tamanhos dos módulos, γ_1;
- número mínimo de vértices por módulo, c_{min};
- número máximo de vértices por módulo, c_{max};
- parâmetro de mistura, μ.

O algoritmo gera redes que seguem aproximadamente os parâmetros fornecidos. O parâmetro de mistura é um número entre 0 e 1 que indica a fração das arestas de cada vértice que são compartilhadas com vértices de outro módulo. Por exemplo, se $\mu = 0,4$, então 40% das arestas de cada vértice são ligadas a vértices que não pertencem ao mesmo módulo do vértice de origem.

Discussão

O modelo gera grafos não-orientados, uma representação muito simplificada das redes de dependências entre classes. Por essa razão ele não é adequado para testar algoritmos de agrupamento que consideram a informação de sentido das arestas. Além disso, ele considera que todos os vértices possuem arestas para vértices de outros módulos, 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).

2.2.3 O modelo XXX

Considerando as limitações do modelo LFR, propomos um novo modelo de rede com estrutura de módulos embutida, baseado no modelo de grafo orientado de Bollobás [2], que foi inspirado na rede de links entre páginas da Web. O modelo possui os seguintes parâmetros:

- número de vértices, N;
- arquitetura, a — uma rede representando os módulos e as ligações permitidas entre módulos;
- probabilidades α, β e γ, tal que $\alpha + \beta + \gamma = 1$;
- probabilidade μ;
- δ_{in} e δ_{out}.

Inicialmente é criada uma rede contendo um vértice com autolaço (aresta ligando o vértice a ele próprio) para cada módulo representado na arquitetura e cada vértice é incluído no módulo correspondente. O algoritmo consiste em alterações sucessivas à rede até se alcançar o número de vértices desejado. Na descrição 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 na rede:

- **Criação de vértice com grau de saída = 1**. Com probabilidade α 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 γ é criado um vértice w juntamente com uma aresta de um vértice 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 β é criada uma aresta de um vértice existente v para um vértice existente w, v escolhido de acordo com $k_{out} + \delta_{out}$ e w escolhido de acordo com $k_{in} + \delta_{in}$. Com probabilidade $1 - \mu$, w é escolhido dentre os vértices do mesmo módulo que v; 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 X_in e X_out, respectivamente. Pode-se demonstrar analiticamente que, quando N tende a infinito, os expoentes podem ser calculados pelas seguintes expressões [2]:

$$X_{in} = 1 + \frac{1+\delta_{in}(\alpha+\gamma)}{\alpha+\beta}$$

$X_{out} = 1 + \frac{\beta+\gamma}{1+\delta_{out}(\alpha+\gamma)}$

Discussão

Este modelo supera as limitações encontradas no modelo LFR: as redes são orientadas, nem todos os vértices são necessariamente ligados a vértices de outros módulos 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 objeto de avaliar a capacidade dos modelos de gerar redes semelhantes a redes de dependências entre classes, realizamos um experimento com dois sistemas, o jogo VilloNanny 2.2.4 e o programa IRPF 2009. O objetivo foi avaliar se, com uma escolha adequada dos parâmetros, os modelos são capazes de gerar redes parecidas às redes dos dois sistemas. O experimento foi dividido em cinco etapas: extração das redes dos sistemas (redes reais), análise das redes reais, escolha dos parâmetros dos modelos, geração de redes sintéticas e comparação entre redes sintéticas e as redes reais correspondentes.

3.1 Extração

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[1].

3.2 Análise de sistemas reais

Baseado em pesquisas anteriores [7, 6, 1], consideramos que a distribuição estatística dos graus dos vértices e a distribuição dos tamanhos dos módulos seguem aproximadamente uma lei de potência. Para cada rede foram coletadas diversas métricas da teoria dos grafos e da teoria das redes complexas:

- número de vértices, N;
- número de arestas, E;
- número de arestas externas E_{ext} — número de arestas que ligam vértices em módulos distintos;
- grau médio, $\langle k \rangle$;
- grau máximo, k_{max};
- expoente da distribuição de graus, γ_g;
- número de módulos, C;
- tamanho do menor módulo, T_{min};
- tamanho do maior módulo, T_{max};
- expoente da distribuição dos tamanhos dos módulos, γ_m

Os expoentes das distribuições de graus e de tamanhos dos módulos foram estimados através da técnica de máxima verossimilhança [3]. Utilizamos uma implementação disponível disponível na Internet[2].

Extraímos ainda a rede de dependências entre MÓDULOS...

3.3 Escolha dos parâmetros dos modelos

Para o modelo de rede aleatória com módulos, mantemos o número de vértices e o número de módulos do sistema. O parâmetro p foi definido para $E/N(N-1)$, de modo que se espera que o grafo produzido pelo modelo tenha E arestas, assim como o sistema.

Os parâmetros do modelo LFR são iguais às métricas correspondentes extraídas dos sistemas.

Para o modelo XXX definimos a arquitetura igual à arquitetura do sistema, número de vértices idem e calculamos $\alpha, \beta, \gamma, \delta_{in}, \delta_{out}$ de forma a satisfazer às seguintes restrições:

- $\gamma = \alpha$. 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

Foram geradas 10 redes para cada modelo. Por que 10?

3.5 Comparação com os sistemas reais

Estatística de Kolmogorov-Smirnov (+qqplot pra ilustrar alguns casos) para comparar... * distribuição de graus * distribuição de coeficiente de clustering * distribuição do número de arestas externas por vértice * distribuição do número de módulos vizinhos por vértice * distribuição dos tamanhos dos módulos

4. RESULTADOS

TABELA 1: sistemas analisados. Nome, versão, métricas (tamanho, número de módulos, ...)

TABELA 2: métricas das redes sintéticas.

TABELA 3 (ou GRÁFICO): distâncias

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

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

5. DISCUSSÃO

Este artigo apresentou uma nova abordagem para avaliação de algoritmos de clustering, através de sistemas sintéticos, e apresentou um modelo de sistema sintético. Esse modelo foi comparado empiricamente a modelos genéricos de redes. NO FINAL: Acreditamos que outras áreas de pesquisa da engenharia de software podem se beneficiar de uma abordagem de avaliação baseada em sistemas de software sintéticos (ex.: análise de impacto, localização de features...).

O quão bem o modelo funciona, vantagens e desvantagens em relação a outras abordagens (complementares).

Especulação sobre o papel da modelagem estatística na engenharia de software.

Focar na hipótese: redes sintéticas com parâmetros ajustáveis dão insights sobre as ferramentas de engenharia reversa / evolução de software.

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

6. REFERENCES

[1] G. Baxter, M. Frean, J. Noble, M. Rickerby, H. Smith, M. Visser, H. Melton, and E. Tempero. Understanding the shape of java software. *SIGPLAN Not.*, 41(10):397–412, 2006.

[2] 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.

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

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

[5] 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.

[6] 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.

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