APIMiner: Uma Plataforma para Recomendação de Exemplos de Uso de APIs

João Eduardo Montandon, Marco Túlio Valente

Departamento de Ciência da Computação, UFMG

{joao.montandon,mtov}@dcc.ufmg.br

Resumo. *Exemplos são fundamentais para facilitar e tornar o uso de APIs mais produtivo. Neste artigo, apresenta-se a ferramenta APIMiner, que extrai exemplos de uso a partir de um repositório privado de sistemas, sumariza os exemplos extraídos usando técnicas de slicing estático de programas e automaticamente incorpora os exemplos sumarizados à documentação, no formato JavaDoc, de uma determinada API alvo. Descreve-se também uma instanciação da solução proposta para incorporar exemplos à documentação de classes da API do sistema operacional Android.*

1 Introdução

APIs (*Application Programming Interfaces*) constituem uma tecnologia para reúso de software amplamente adotada, com diversos casos práticos de sucesso. Por exemplo, um dos motivos frequentemente apontados para o sucesso da linguagem Java é a existência de uma API padrão, contendo mais de três mil classes e 27 mil métodos.

No entanto, o aprendizado de uma API pode se revelar uma tarefa não-trivial, conforme apontado em estudos empíricos recentes [5]. Conforme indicado por tais estudos, um dos principais obstáculos para o uso de APIs está relacionado com a qualidade da documentação disponível, a qual normalmente se limita a descrever a assinatura dos métodos disponibilizados pela API. Mais precisamente, ainda não existe um consenso sobre como agregar exemplos de uso em documentos que descrevem APIs. A existência de tais exemplos é considerada um recurso chave para facilitar e tornar mais produtivo o uso de APIs, principalmente por parte de programadores não familiarizados com as mesmas.

Neste artigo, apresenta-se a plataforma APIMiner para agregação automática de exemplos de uso em documentações tradicionais de APIs para a linguagem Java, disponibilizadas no formato JavaDoc. A plataforma para recomendação de exemplos proposta possui quatro características principais:

- Exemplos de uso são extraídos automaticamente a partir de um repositório privado de sistemas, que deve ser configurado pelos administradores que utilizarão a plataforma APIMiner. Em outras palavras, exemplos não são extraídos de programas publicamente disponíveis na Web, tal como ocorre com outras plataformas para recomendação de APIs, como Koders [1] e APIExample [6]. Em geral, quando se utiliza código público, a tarefa de determinar a relevância do código recomendado se torna mais complexa.

- Exemplos de uso são sumarizados por meio de um algoritmo simplificado de *slicing* estático [7], a fim de disponibilizar fragmentos de código mais relevantes para entendimento de um determinado método da API. Assim, procura-se disponibilizar não apenas o trecho de código exato onde a API foi usada, mas também o contexto sintático onde esse uso ocorreu. Esse contexto inclui as dependências de dados e de controle que impactam diretamente a execução do comando responsável pelo uso da API.

- Exemplos de uso são classificados e ordenados usando informações disponibilizadas em um sistema de controle de versões. A premissa é que tais informações são mais úteis para encontrar exemplos relevantes do que informações de acoplamento, usadas por algoritmos clássicos de análise de *links*, como PageRank [2]. Atualmente, a ferramenta considera que exemplos provenientes de arquivos com grande número de *commits* no sistema de controle de versões são mais relevantes.

- Exemplos de uso são automaticamente agregados na documentação no formato JavaDoc da API alvo. Assim, a fase de extração, sumarização, classificação e instrumentação dos arquivos JavaDoc ocorre de modo *off-line*. Durante a fase de consulta, os exemplos gerados são automaticamente recuperados de um banco de dados, sem necessidade de processamento extra.

Além de descrever o funcionamento da plataforma APIMiner, esse artigo apresenta um exemplo de uso da solução proposta, envolvendo a geração de exemplos para classes da API do sistema operacional Android. Para essa instanciação particular da solução proposta, denominada Android APIMiner, foi criado um repositório com 22 programas bem conhecidos para Android e incorporados exemplos em documentos JavaDoc de quatro classes dessa API.

O restante deste artigo está organizado conforme descrito a seguir. A Seção 2 apresenta uma visão geral da solução proposta pela plataforma APIMiner. A Seção 3 descreve o exemplo de uso para o sistema operacional Android. A Seção 4 compara a solução proposta com outras ferramentas. A Seção 5 conclui o trabalho.

2 APIMiner: Visão Geral

O funcionamento da plataforma APIMiner é dividido em duas fases: pré-processamento e consulta. A fase de pré-processamento é responsável por realizar todo o processamento necessário para obtenção dos exemplos, desde o *parser* dos arquivos até a sumarização, ranqueamento e armazenamento dos exemplos. A fase de consulta contempla a interação do usuário com a ferramenta. Basicamente, nessa fase o usuário pode acessar o JavaDoc gerado pela plataforma e buscar na base de dados pelos exemplos de uso desejados.

A Figura 1 apresenta a arquitetura interna da plataforma, a qual inclui seis componentes principais:

Lista de Métodos da API: Uma base de dados que possuirá a lista de todos os métodos da API para a qual se deseja obter exemplos de uso. A solução proposta

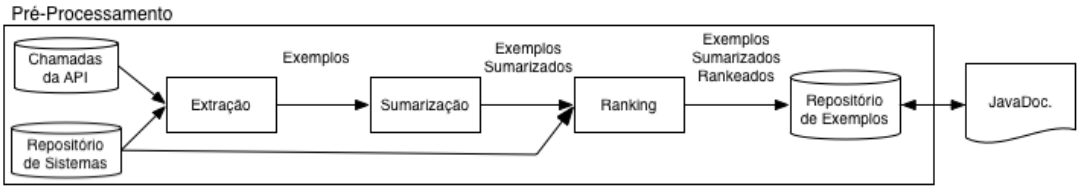

Figura 1. Arquitetura da plataforma APIMiner

irá disponibilizar exemplos para o conjunto de métodos informados nessa lista.

Repositório de Sistemas: Repositório com o código fonte dos sistemas que serão usados para extração dos exemplos a serem disponibilizados pela plataforma.

Módulo de Extração: Esse módulo é responsável pela primeira etapa de processamento da plataforma. Nessa etapa, o APIMiner procura no Repositório de Sistemas por chamadas a métodos cadastrados na Lista de Métodos da API. Para isso, esse módulo faz o *parser* de cada arquivo do Repositório de Sistemas e analisa as chamadas de métodos existentes nesses arquivos. Caso uma dessas chamadas seja de um método da API, o corpo do método onde a chamada ocorre é repassado para o Módulo de Sumarização. Para implementação do *parser*, foi utilizado o compilador Eclipse Compiler for Java (ECJ) do componente JDT da plataforma Eclipse.

Módulo de Sumarização: Esse módulo recebe como entrada o corpo de um método `g` contendo pelo menos uma chamada a um método `f` da API. O módulo então analisa o corpo de `g` a fim de retornar apenas aqueles comandos, declarações e expressões importantes para compreensão da chamada de `f`. Para isso, foi desenvolvido um algoritmo de sumarização de exemplos baseado em princípios de *slicing* estático de programas, o qual será apresentado com mais de detalhes na Seção 2.1.

Módulo de Ranqueamento: Esse método ordena os exemplos extraídos e sumarizados pelos módulos anteriores, usando informações de gerenciamento de projetos disponibilizadas no sistema de controle de versões. Na versão atual da ferramenta, os exemplos são ordenados em função do número de *commits* do arquivo de onde foram extraídos. A premissa é que se um arquivo é frequentemente modificado, ele é responsável por uma funcionalidade importante do sistema alvo e, portanto, possui bom potencial para gerar exemplos interessantes e relevantes.

Repositório de Exemplos: Base com os exemplos gerados pela sumarização.

Interface JavaDoc: Interface para comunicação com a plataforma.

2.1 Algoritmo para Sumarização de Exemplos

Conforme afirmado, esse algoritmo baseia-se em conceitos de *slicing* estático [7]. Para facilitar sua apresentação, considere que deseja-se obter um exemplo para o método `substring` e que o Módulo de Extração tenha encontrado no Repositório de Sistemas a seguinte chamada desse método:

```
String r= client.substring(ini, end);   // semente da sumarização
```

Esse comando é denominado de semente do algoritmo de sumarização (isto é, ele inclui a chamada para a qual se deseja extrair comandos adjacentes relevantes

para seu entendimento). Como pode ser observado, somente esse comando não carrega informações suficientes para exemplificar o uso do método `substring(int,int)`. Para entendê-lo melhor, é importante conhecer também os valores das variáveis `client`, `ini` e `end`. Adicionalmente, é importante saber como o valor retornado pela chamada do método (no caso, armazenado em `r`) será utilizado posteriormente. Por fim, é interessante conhecer as dependências de controle (devido a comandos de repetição e decisão) que influenciam a execução dessa chamada.

Em outras palavras, é preciso determinar os seguintes conjuntos: (C1) comandos anteriores que definem valores para as variáveis `client`, `ini` e `end`, (C2) comandos posteriores que utilizam a variável `r` e (C3) dependências de controle que interferem na execução da chamada do método. Para isso, foi desenvolvido um algoritmo que extrai as dependências de dados e controle de uma semente e determina os conjuntos C1, C2 e C3. Por exemplo, a aplicação do algoritmo sobre a chamada apresentada anteriormente poderia gerar o seguinte exemplo:

```
01: String client= "Smith, John";                    // define client
05: int ini= 0;                                      // define ini
06: int end= client.indexOf(",");                    // define end
20: if (end >= 0) {                                  // testa end
21:   String r= client.substring(ini, end);          // semente
22:   System.out.println("Welcome back Mr." + r);    // usa r
23: }
```

3 Exemplo de Uso: Android APIMiner

Para avaliação preliminar e realização de testes funcionais com a ferramenta, foi construído um exemplo de uso envolvendo a API Android. Para isso, foram coletados exemplos relativos a chamadas de métodos de quatro classes da API, descritas na Tabela 1. Ainda, 22 sistemas de código aberto foram baixados e armazenados no Repositório de Sistemas.

Tabela 1. Classes da API Android consideradas no exemplo

Classe	Descrição
Toast	Exibição de mensagens de notificação para os usuários
Vibrator	Manipulação do dispositivo responsável pela vibração do equipamento
BluetoothAdapter	Manipulação do adaptador Bluetooth
TextView	Componente de interface para manipulação de texto

Durante a fase de pré-processamento, foram considerados 252 métodos das classes mencionadas na Tabela 1. Como resultado, a ferramenta foi capaz de extrair 1832 exemplos, cobrindo 43 métodos diferentes (isto é, cerca de 17% dos métodos considerados na experiência). Para ilustrar um dos exemplos gerados pela ferramenta APIMiner, será usado o método `setGravity` da classe `android.widget.Toast`. Basicamente, esse método define o posicionamento de um objeto do tipo `Toast` na tela, o qual denota uma mensagem em forma de *pop-up*. A documentação oficial desse método, mostrada na Figura 2, é bastante sucinta. Ela não descreve, por exemplo, como os parâmetros de entrada do método podem ser obtidos.

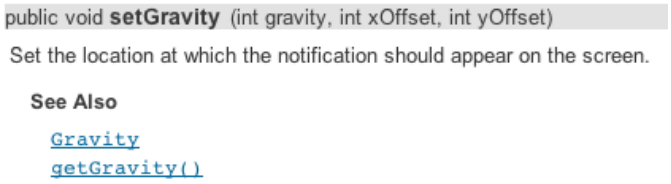

Figura 2. Documentação do método setGravity

Para esse método, a ferramenta APIMiner produziu o seguinte exemplo:

```
456: String storageState= Environment.getExternalStorageState();
457: if (!storageState.equals(Environment.MEDIA_MOUNTED)) {
459:   Toast toast= Toast.makeText(this, R.string.storage_warning,
                                  Toast.LENGTH_LONG);
461:   toast.setGravity(Gravity.CENTER,0,0);
463: }
```

Esse código disponibiliza informações importantes para utilização do método `setGravity`, tais como a forma de se obter um objeto alvo do tipo `Toast` (linha 459), a existência de constantes que podem ser usadas para centralizar a mensagem na tela (linha 461) e que esse exemplo, em particular, está relacionada com o fato de um dispositivo externo estar montado ou não (linhas 456 e 457). Pode-se verificar ainda que as linhas do exemplo não são consecutivas, isto é, o Módulo de Sumarização eliminou do exemplo algumas linhas de código intermediárias que não possuem dependências de dados ou de controle com a semente considerada (linha 461).

4 Ferramentas Relacionadas

Ferramentas relacionadas se dividem em três grupos, conforme apresentado na Tabela 2. O primeiro grupo inclui máquinas de busca, como Koders [1] e APIExample [6], as quais em geral oferecem exemplos sem nenhuma forma de sumarização. O segundo grupo inclui sistemas de recomendação que funcionam como *plug-ins* de ambientes integrados de desenvolvimento, como Strathcona [3]. A vantagem nesse caso é que esses sistemas podem se valer do contexto sintático onde um determinado método da API deve ser chamado para oferecer exemplos mais relevantes. Por outro lado, esse grupo de sistemas não é tão útil para programadores que se encontram em uma fase inicial do aprendizado de uma API. Por fim, existem sistemas que instrumentam a documentação de uma API com exemplos de uso, tal como a ferramenta descrita neste artigo (APIMiner). Conforme apresentado na Tabela 2, a ferramenta eXoaDocs [4] é a mais próxima da ferramenta proposta neste artigo. No entanto, o eXoaDocs não dispõe de um repositório próprio de sistemas (que permitam a geração de exemplos mais relevantes e confiáveis). Em vez disso, o eXoaDocs usa como entrada exemplos fornecidos por uma máquina de busca (mais especificamente, pelo sistema Koders). Além disso, o eXoaDocs apenas considera dependências de dados para sumarização dos exemplos providos pela máquina de busca Koders.

5 Comentários Finais

Neste artigo, foi apresentada a ferramenta de recomendação APIMiner, que se vale de um repositório de dados privados para associação de exemplos à documentação

Tabela 2. Comparação com ferramentas relacionadas

Categoria	Sistema	Interf.	Entrada	Repos.	Sumariz.
Máquina de busca	Koders	Web	Texto	Público	Não
	APIExample	Web	Tipo	Público	Não
Recomendação tempo de desenvolvimento	Strathcona	IDE	Comando	Privado	Não
Recomendação associada a documentação	eXoaDocs	JavaDoc	Método	—	Slicing dados
	APIMiner	JavaDoc	Método	Privado	Slicing dados e controle

tradicional de APIs usando o formato JavaDoc. Os exemplos gerados são sumarizados de forma *off-line* usando técnicas de *slicing* estático que consideram tanto dependências de dados como de controle. Os exemplos são classificados usando informações de gerência de projetos, extraídas de um sistema de controle de versões. Um estudo de caso foi apresentado, envolvendo a API do sistema operacional Android. Atualmente, estamos trabalhando no projeto de um experimento para avaliar com programadores a relevância dos exemplos providos pela ferramenta APIMiner.

A ferramenta APIMiner configurada para uso com a API do sistema operacional Android está disponível em: `http://java.llp.dcc.ufmg.br/apiminer`.

Agradecimentos: Este trabalho foi apoiado pela FAPEMIG e CNPq.

Referências

[1] Koders. http://koders.com.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 2011.

[3] Reid Holmes, Robert J. Walker, and Gail C. Murphy. Approximate structural context matching: An approach to recommend relevant examples. *IEEE Transactions on Software Engineering*, 32(12):952–970, 2006.

[4] Jinhan Kim, Sanghoon Lee, Seung-won Hwang, and Sunghun Kim. Towards an intelligent code search engine. *24th AAAI Conference on Artificial Intelligence*, pages 1358–1363, 2010.

[5] Martin P. Robillard and Robert DeLine. A field study of API learning obstacles. *Empirical Software Engineering*, 16(6):703–732, 2010.

[6] Leye Wang, Lu Fang, and Ge Li. APIExample: An effective web search based usage example recommendation system for Java APIs. *26th Automated Software Engineering Conference (ASE)*, pages 592–595, 2011.

[7] Mark Weiser. Program slicing. In *5th International Conference on Software Engineering (ICSE)*, pages 439–449, 1981.