Redocumentando APIs com Conhecimento da Multidão: um estudo de cobertura da API Swing no Stack Overflow

Fernanda M. Delfim[1], **Klérisson V. R. da Paixão**[1], **Marcelo de A. Maia**[1]

[1]LASCAM – Laboratory of Software Comprehension, Analytics and Mining
Universidade Federal de Uberlândia (UFU)

{fernanda,klerisson}@doutorado.ufu.br, marcelo.maia@ufu.br

Abstract. *API documentation, when exists, is insufficient to assist developers in their specific tasks. Few people write the API documentation for many potential readers, resulting in the lack of explanations and examples about different scenarios and perspectives. In this paper, we report a feasibility study to use the knowledge generated by the crowd on Stack Overflow for API documentation. The focus is to measure the contribution of the crowd with snippets about elements of an API in answers of how-to-do-it questions, in which the questioner provides a scenario and asks about how to implement it. We found that more than half of the Swing classes are not covered in questions-answers of that type, claiming for new incentive mechanisms for uncovered parts of the API.*

Resumo. *A documentação de API, quando existe, é insuficiente para auxiliar desenvolvedores em suas tarefas específicas. Poucas pessoas escrevem a documentação de API para muitos potenciais leitores, resultando na falta de explicações e exemplos sobre diferentes cenários e perspectivas. Neste trabalho é relatado um estudo de viabilidade em utilizar o conhecimento gerado pela multidão no Stack Overflow para a documentação de API. O foco é medir a contribuição da multidão com trechos de código sobre elementos de uma API em respostas de perguntas how-to-do-it, em que o questionador provê um cenário e pergunta sobre como implementá-lo. Foi observado que mais da metade das classes do Swing não estão cobertas em perguntas-respostas desse tipo, o que indica a necessidade de novos mecanismos de incentivo para partes não cobertas da API.*

1. Introdução

Desenvolvedores precisam de conhecimento contínuo sobre as tecnologias e ferramentas que utilizam durante a implementação de sistemas de software. A documentação dessas tecnologias e ferramentas, no entanto, normalmente não existe ou é insuficiente. Considerando API (*Application Programming Interface*), em [Robillard 2009] foi realizado um *survey* com 80 desenvolvedores da Microsoft sobre os obstáculos na aprendizagem de API. Como resultado, foi descoberto que no topo da lista de obstáculos aparecem os que são causados por recursos ausentes ou inadequados como, por exemplo, documentação.

O sucesso das mídias sociais introduziu novas maneiras de troca de conhecimento por meio da Internet [Treude et al. 2011]. Wikis, blogs e sites de perguntas e respostas (Q&A) são novas maneiras de contribuição e colaboração que têm o potencial de redefinir como os desenvolvedores aprendem, preservam e comunicam conhecimento sobre desenvolvimento de software [Parnin et al. 2012]. A informação disponível

nesse tipo de serviço é conhecida como *conhecimento da multidão* (*crowd knowledge*) [Souza et al. 2014b], em que a multidão é formada por indivíduos que contribuem com conhecimento de diferentes maneiras. No site de Q&A Stack Overflow (SO)[1], por exemplo, os indivíduos podem contribuir fazendo ou rotulando uma pergunta, fornecendo ou votando em uma resposta, entre outras ações.

O conhecimento da multidão disponível nas mídias sociais pode ser utilizado para documentar APIs. O resultado de tal documentação é chamado de *documentação pela multidão* (*crowd documentation*), isto é, conhecimento que é escrito por muitos e lido por muitos [Parnin et al. 2012][Souza et al. 2014a]. Nesse contexto, estudos sobre a viabilidade de utilizar o conhecimento da multidão para documentar APIs são necessários. Parnin et al. [Parnin et al. 2012] realizaram um estudo sobre tal viabilidade por meio da análise de cobertura de elementos de API pela multidão em threads (uma pergunta com nenhuma ou várias respostas) do SO. No entanto, nenhum critério foi estabelecido sobre os tipos de pergunta, de postagem (pergunta e resposta) e de conteúdo (texto e trecho de código, por exemplo) para a análise de cobertura. Para exemplificar a necessidade do estabelecimento de critérios para a seleção dos dados a fim de analisar a cobertura, considere as threads que possuem pergunta do tipo *Debug/Corrective* [Nasehi et al. 2012] e *Review* [Treude et al. 2011], em que são apresentados problemas e trechos de código com defeito. O conteúdo dessas threads não é necessariamente útil para a documentação de API e, portanto, não deveria ser utilizado para medir a cobertura de elementos de API.

Neste trabalho é relatado um estudo sobre a viabilidade de utilizar o conhecimento da multidão do SO para a documentação de API, onde a API Swing foi utilizada como estudo de caso por ser amplamente utilizada e bem consolidada. O principal objetivo é medir a cobertura de elementos da API para analisar como a multidão contribui para a documentação de API com *exemplos de trechos de código* de *como realizar uma determinada tarefa*. Um classificador foi utilizado para a seleção de threads com pergunta do tipo *how-to-do-it*, em que o questionador provê um cenário e pergunta sobre como implementá-lo. Exemplos de código sobre tal cenário são fornecidos nas respostas, sendo estes potencialmente úteis para a documentação de API.

O restante deste trabalho está organizado como segue. Os trabalhos relacionados são citados na Seção 2. Na Seção 3 é descrita a metodologia do estudo, que inclui os objetivos, as questões de pesquisa e a coleta e análise de dados. Os resultados do estudo são apresentados na Seção 4. Na Seção 5 são apresentadas as limitações deste trabalho, e as conclusões e os trabalhos futuros são expostos na Seção 6.

2. Trabalhos Relacionados

Nasehi et al. [Nasehi et al. 2012] definiram que os tipos de pergunta que podem ser feitas no SO podem ser descritos com base em duas dimensões: 1) o tópico da pergunta, como a principal tecnologia que a pergunta envolve, e 2) os principais interesses do questionador. Eles definiram 4 categorias sobre os interesses do questionador. Treude et al. [Treude et al. 2011] também definiram categorias sobre tais interesses, e descobriram que o SO é eficaz em revisões de código (tipo de pergunta *review*), para perguntas conceituais (*conceptual*) e para os novatos (*novice*), e os tipos de perguntas mais frequentes incluem perguntas *how-to* e perguntas sobre comportamentos inesperados (*discrepancy*).

[1] `http://www.stackoverflow.com`

A identificação automática do tipo de pergunta a partir da primeira dimensão pode ser realizada pela análise dos rótulos (*tags*) atribuídos para a pergunta. Sobre a identificação do tipo de pergunta a partir da segunda dimensão, que envolve os principais interesses do questionador, Souza et al. [Souza et al. 2014b] conduziram um estudo experimental para a classificação automática de pares de Q&A em três categorias: *how-to-do-it*, *conceptual* e *seeking-something*. Uma comparação entre diferentes algoritmos de classificação foi realizada e os melhores resultados foram obtidos com um classificador de regressão logística com taxa de sucesso global de 76.19% e 79.81% na categoria *how-to-do-it*.

Parnin e Treude [Parnin e Treude 2011] realizaram um estudo sobre como os métodos da API jQuery são documentados na Web por meio da análise dos 10 primeiros resultados retornados por pesquisas usando o Google. Eles descobriram que, exceto a documentação oficial da API, as duas fontes que mais cobrem os métodos da API são de mídias sociais: blogs de desenvolvimento de software com 87.9% e SO com 84.4% de métodos cobertos. Jiau e Yang [Jiau e Yang 2012] replicaram o estudo de Parnin e Treude para três APIs orientadas a objetos e descobriram que a cobertura de métodos pelo SO para essas APIs é consideravelmente menor do que no estudo original (84.4% para jQuery), sendo 29.11%, 59.19%, 37.64% para Swing, GWT e SWT, respectivamente.

O trabalho mais relacionado com este é o trabalho de Parnin et al. [Parnin et al. 2012], em que foi relatado um estudo empírico sobre a viabilidade de utilizar o conhecimento da multidão do SO para a documentação de três APIs (Java, Android e GWT). Para tanto, eles construíram um modelo de rastreabilidade entre elementos de API (classes) e threads do SO em que esses elementos são citados, e calcularam a porcentagem de elementos que têm ligação com alguma thread, resultando na cobertura de classes. Uma cobertura alta sugeriria que é viável utilizar o conhecimento da multidão para a documentação de API como uma fonte abrangente de conhecimento sobre os elementos de tal API. Eles descobriram que 77.3% e 87.2% das classes das APIs Java e Android, respectivamente, são cobertas pela multidão, e concluíram que apesar do potencial da documentação pela multidão em prover muitos exemplos e explicações sobre elementos de API, a multidão não é confiável para fornecer Q&A para uma API inteira. As principais diferenças entre este trabalho e o trabalho de Parnin et al. são:

Threads analisadas. Em [Parnin et al. 2012], todas as threads com pergunta rotulada com o nome da API sob investigação são utilizadas para a análise de cobertura. Neste trabalho, além disso, somente as threads com pergunta do tipo *how-to-do-it* são utilizadas.

Conteúdo de thread analisado. Em [Parnin et al. 2012], todo o conteúdo textual de postagens foi analisado. Neste trabalho, somente trechos de código foram analisados, pois o foco do trabalho é verificar a existência de exemplos de uso de elementos de API.

Tipo de postagem analisada. Em [Parnin et al. 2012], os dois tipos de postagem existentes (pergunta e resposta) foram analisados. Visto que o foco deste trabalho são threads com pergunta do tipo *how-to-do-it*, somente respostas foram analisadas.

Tipo de elemento de API analisado. Em [Parnin et al. 2012], o tipo de elemento de API analisado foi classe. Neste trabalho, classes, interfaces e enumerações foram analisadas.

3. Metodologia do Estudo

Os *objetivos* deste estudo são: 1) *analisar* discussões sobre a API Swing no SO, *com o propósito de* investigar a viabilidade de utilizar o conhecimento da multidão para documentação de API, *com respeito a* cobertura de elementos da API por trechos de código em respostas de perguntas do tipo *how-to-do-it*, e 2) *analisar* os resultados de cobertura obtidos juntamente com os resultados obtidos usando a abordagem de Parnin et al. [Parnin et al. 2012], *com o propósito de* comparar os resultados de cobertura, *com respeito aos* filtros aplicados pela abordagem proposta neste estudo. Os dois objetivos são *do ponto de vista de* pesquisadores interessados em utilizar o conhecimento provido pela multidão na documentação de APIs, *no contexto de* 38134 threads relacionadas ao Swing da versão de 21 de Janeiro de 2014 dos dados públicos do SO.

3.1. Questões de Pesquisa

A fim de alcançar os objetivos apresentados, duas questões de pesquisa foram formuladas:

RQ_1: Qual é a proporção de elementos do Swing cobertos pela multidão no Stack Overflow em trechos de código de respostas de perguntas do tipo how-to-do-it?

RQ_2: Qual é a diferença de cobertura em analisar todo o conteúdo de perguntas e respostas de todas as threads e analisar somente trechos de código em respostas de perguntas do tipo how-to-do-it?

3.2. Coleta de Dados

Os dados necessários para responder às questões de pesquisa são 1) as threads relacionadas ao Swing do SO e 2) uma lista de classes, interfaces e enumerações existentes no Swing. As threads do SO utilizadas neste estudo são da versão de 21 de Janeiro de 2014 dos dados públicos do SO, e foram baixadas do *Stack Exchange Data Dump*[2] e importadas em um banco de dados relacional. A lista de elementos existentes no Swing foi obtida pela especificação oficial da API[3], sendo 823 classes, 90 interfaces e 10 enumerações distribuídas em 18 pacotes.

3.3. Análise de Dados

A fim de analisar a cobertura de elementos do Swing em threads do SO, primeiramente foi feita a seleção das threads com rótulo "swing" por meio de correspondência exata de palavra, totalizando 38134 threads.

A análise de cobertura é realizada da seguinte maneira. Para cada thread do nosso conjunto de threads sob análise é procurado em seu conteúdo a ocorrência de citação de nome de classes, interfaces e enumerações do Swing. Quando a citação de nome de um elemento é encontrada, uma ligação entre o elemento e a thread é criada. Como resultado, cada elemento da API terá uma lista de threads em que seu nome foi citado.

A abordagem de análise de cobertura deste trabalho possui alguns filtros sobre os dados: somente threads com pergunta do tipo *how-to-do-it*, respostas como tipo de postagem e trechos de código como conteúdo de thread são analisados. Visto que um dos objetivos deste trabalho é comparar a abordagem deste com a abordagem de Parnin et

[2] `https://archive.org/details/stackexchange`
[3] `http://docs.oracle.com/javase/8/docs/api`

al. [Parnin et al. 2012], a análise de cobertura foi realizada, como explicado no parágrafo anterior, para 8 diferentes configurações: com os três filtros mencionados (abordagem deste trabalho), sem os três filtros (abordagem de Parnin et al.), e outras seis configurações com a combinação dos filtros (veja a primeira coluna da Tabela 1).

Para as análises de cobertura com configurações que utilizam somente threads com pergunta do tipo *how-to-do-it* foi utilizado o classificador de regressão logística de Souza et al. [Souza et al. 2014b] para a identificação das threads de interesse. Como o resultado do classificador são pares de Q&A, as threads foram reconstruídas em perguntas com suas respostas usando o identificador da thread. Como pode ser observado na segunda coluna da Tabela 1, 65.75% das 38134 threads relacionadas ao Swing possuem pergunta classificada como *how-to-do-it*. Para as análises de cobertura com configurações que utilizam somente trechos de código como tipo de conteúdo de thread, para cada postagem, todos os blocos de código foram coletados de dentro da *tag* `<code>`.

4. Resultados

RQ_1: Qual é a proporção de elementos do Swing cobertos pela multidão no Stack Overflow em trechos de código de respostas de perguntas do tipo how-to-do-it?

Para responder a RQ_1, a análise de cobertura foi realizada sobre as 25073 threads que possuem pergunta classificada como *how-to-do-it*, em que somente trechos de código de respostas foram analisados. Tal análise é indicada pelo número 1 na Tabela 1. Somente 51.22% das threads analisadas foram ligadas com algum elemento do Swing.

Sobre a cobertura de elementos, para 44.23% das classes, 87.78% das interfaces e 70% das enumerações do Swing existe pelo menos uma resposta de pergunta *how-to-do-it* com trecho de código no SO. Sobre as classes, 43.38% do total de classes do Swing são classes internas. Foi observado que somente 17.37% delas foram cobertas e que 64.27% das classes não cobertas são classes internas. Para interfaces e enumerações foi observado que a porcentagem de elementos internos não cobertos é ainda maior. Para interface, 33.33% das internas foram cobertas e 72.73% das interfaces não cobertas são internas. Para enumeração, 62.5% das internas foram cobertas e 100% das enumerações não cobertas são internas.

Analisando a cobertura de elementos intra-pacote usando o mapa de calor apresentado na Figura 1 foi observado que 3 pacotes (de um total de 18) foram totalmente cobertos: `javax.swing.filechooser`, `javax.swing.text.rtf` e `javax.swing.border` (representados pelos três retângulos menores em vermelho). 2 desses pacotes, no entanto, estão no top 5 pacotes com menos elementos. O pacote com mais elementos, `javax.swing`, sendo 209 classes, 25 interfaces e 7 enumerações, obteve 60.29%, 92% e 85.71% de seus elementos cobertos, respectivamente. Além disso, nenhum elemento do pacote `javax.swing.plaf.multi` (representado pelo retângulo em cinza claro) foi coberto.

RQ_2: Qual é a diferença de cobertura em analisar todo o conteúdo de perguntas e respostas de todas as threads e analisar somente trechos de código em respostas de perguntas do tipo how-to-do-it?

Para responder a RQ_2 foi realizada a análise de cobertura sem os filtros propostos neste trabalho, sendo assim o trabalho de Parnin et al. replicado (análise indicada por 2

na Tabela 1). Além disso, seis análises de cobertura complementares foram realizadas, em que os filtros foram combinados a fim de observar quais filtros contribuem mais para a diminuição de cobertura (análises indicadas de 3 a 8 na Tabela 1).

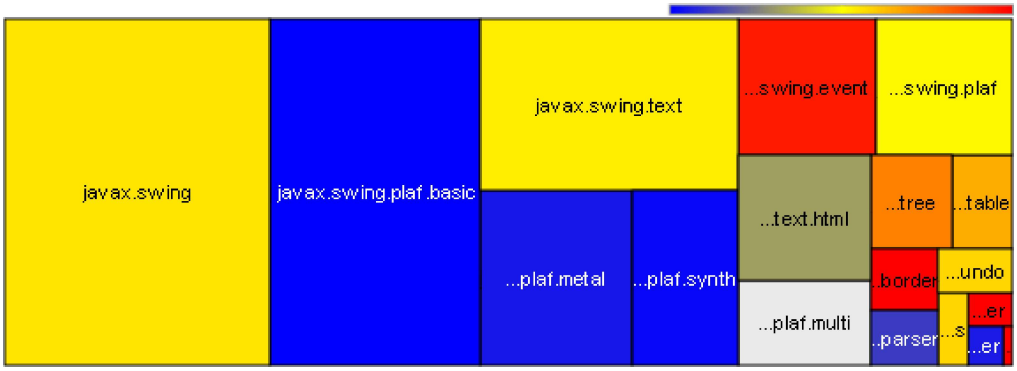

Figura 1. Mapa de calor sobre a cobertura intra-pacote, considerando classes, interfaces e enumerações. A escala de cor representa valores percentuais de cobertura sobre os elementos, sendo que azul representa o valor mais baixo e vermelho o valor mais alto. O tamanho de cada retângulo representa proporcionalmente a quantidade de elementos do pacote.

Comparando diretamente a abordagem deste trabalho com a abordagem de Parnin et al. (análises 1 e 2, respectivamente) foi observado que, apesar de que com a análise 2 mais do que o dobro de threads tenham sido ligadas com algum elemento do Swing, apenas 12.88% a mais das classes foram cobertas (57.11%), o que ainda mantém a cobertura de classes baixa. A cobertura de interfaces é razoavelmente alta na análise 1 (87.78%), e aumentou apenas 1.11% na análise 2. Para enumerações, a cobertura aumentou em 20% na análise 2, mas vale ressaltar que, observando os valores absolutos, somente 2 enumerações a mais foram cobertas.

Tabela 1. Configurações e resultados da análise de cobertura. Na primeira coluna são descritas as 8 configurações, em que três parâmetros foram combinados: tipo de pergunta (*how-to-do-it* ou todos), tipo de postagem (resposta ou Q&A) e tipo de conteúdo de thread (trecho de código ou todos).

Configuração (tipo de <>)	**Medidas sobre threads**		**Medidas sobre cobertura**		
<pergunta>, <postagem>, <conteúdo>	#Threads	#Ligadas	Classes	Interfaces	Enums
1) how-to, resposta, código	25073	12843	44.23%	87.78%	70%
2) todos, Q&A, todos	38134	27660	57.11%	88.89%	90%
3) how-to, Q&A, todos	25073	18686	53.58%	88.89%	80%
4) todos, resposta, todos	38134	27660	52.73%	88.89%	90%
5) todos, Q&A, código	38134	17993	54.07%	88.89%	80%
6) how-to, resposta, todos	25073	18686	49.45%	88.89%	80%
7) how-to, Q&A, código	25073	12843	49.70%	88.89%	70%
8) todos, resposta, código	38134	17993	47.51%	87.78%	80%

Analisando os resultados das análises de 3 a 8 foi observado que quando os filtros são aplicados separadamente (análises 3, 4 e 5) a cobertura de elementos diminui menos em comparação com as análises em que dois filtros são aplicados em conjunto (6, 7 e 8). Isso sugere que não existe uma diferença significativa em termos de cobertura para concluir que um filtro diminui mais a cobertura do que os outros, porque nenhum filtro aplicado sozinho diminui mais a cobertura do que os outros dois aplicados em conjunto.

Por exemplo, suponha que com o filtro de tipo de pergunta (somente *how-to* são analisadas) tenha sido obtida uma cobertura menor do que com os filtros de tipo de postagem (somente respostas são analisadas) e de conteúdo de thread (somente código é analisado) aplicados em conjunto. Dessa maneira seria possível concluir que o filtro de tipo de pergunta é o que mais contribui para a abordagem deste trabalho ter coberto 12.88% a menos das classes em comparação com a abordagem de Parnin et al.

Apesar de não existir uma diferença significativa em termos de cobertura para concluir que um filtro diminui mais a cobertura do que os outros, os resultados podem indicar qual filtro *não* é o que mais diminui a cobertura. Analisando os resultados das análises que combinam dois filtros (6, 7 e 8), foi observado que, para classes, as combinações do filtro *how-to* com os outros dois filtros (análises 6 e 7) produziram uma cobertura semelhante, diminuindo 4.13% de classes cobertas em comparação com a sua aplicação com nenhum outro filtro (análise 3), e para interfaces a cobertura foi mantida em 88.89%. Então, a pior cobertura para classes e interfaces é quando a análise de cobertura é feita sobre código em respostas de todas as threads (análise 8), indicando que o filtro de tipo de pergunta *how-to* não é o mais prejudicial para a diminuição de cobertura.

Adicionalmente, foi observado que com os pares de análises que variam somente o tipo de postagem – (1,7), (2,4), (3,6) e (5,8) – a mesma quantidade de threads são ligadas com elementos do Swing. Por exemplo, com as análises 1 e 7, 51.22% das threads foram ligadas com algum elemento, mesmo que a cobertura da análise 1 (com o filtro de tipo de postagem) tenha sido menor do que da análise 7 (sem o filtro). Isso sugere que as threads ligadas com o filtro são as mesmas ligadas sem o filtro, e que nas perguntas são citados elementos diferentes do que em suas respostas.

5. Limitações

Na análise de cobertura, o conteúdo de threads é analisado para a identificação de nomes de elementos de API por correspondência exata. Sendo assim, em alguns casos, citações de elementos podem ser perdidas. Por exemplo, se a classe `javax.swing.JFrame` foi citada por engano com o "f" em minúsculo, tal citação não será encontrada. Além disso, elementos que possuem o mesmo nome curto não são ligados com threads em que possuem citação somente do nome curto. No Swing, o elemento `Element`, por exemplo, foi citado em trechos de código de 66 threads, mas tais citações não foram contabilizadas para a classe `javax.swing.text.html.parser.Element` e para a interface `javax.swing.text.Element`. Ainda assim, a interface foi coberta por 9 threads em que seu nome completo foi citado, mas a classe ficou sem cobertura. Mesmo que em tal análise somente trechos de código tenham sido analisados, uma alternativa poderia ser analisar o texto da postagem na tentativa de resolver colisão de nomes.

Neste trabalho foram analisadas classes, interfaces e enumerações como tipo de elemento de API. Os resultados para esses elementos, no entanto, não podem ser generalizados para outros, como pacotes e métodos. Além disso, somente uma API (Swing) foi analisada, sendo uma limitação à generalização dos resultados para outras APIs.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentado um estudo sobre a cobertura de elementos da API Swing pela multidão no SO, com foco em threads com perguntas do tipo *how-to-do-it*, pois estas

são consideradas pertinentes para a documentação de API. Foi descoberto que a multidão provê trechos de código para apenas 44.23% de classes, 87.78% de interfaces e 70% de enumerações do Swing. A cobertura de classes não ultrapassou a metade da quantidade de classes existentes no Swing, e mesmo que para interfaces e enumerações a cobertura tenha sido razoavelmente alta, ainda faltam muitos elementos a serem cobertos pela multidão.

A abordagem proposta e utilizada neste trabalho, de considerar apenas trechos de código fornecidos em respostas de perguntas *how-to-do-it*, foi comparada com a abordagem de Parnin et al., em que nenhum critério sobre os dados foi estabelecido para analisar a cobertura de elementos de API. Os resultados mostraram que, com a ausência de filtros, mais do que o dobro de threads foram ligadas com algum elemento do Swing, mas apenas 12.88% a mais das classes foram cobertas, 1.11% das interfaces e 20% das enumerações (2 elementos), o que ainda mantém uma parte grande do Swing descoberta. Vale ressaltar que a quantidade de threads ligadas com elementos de API não é necessariamente importante. O importante é o conteúdo pertinente para a documentação de API, e por esse motivo neste trabalho foi proposta a utilização de threads com pergunta *how-to-do-it*.

Como trabalhos futuros, outros tipos de elementos de API, como pacotes e métodos, devem ser analisados, bem como a cobertura para outras APIs. Além disso, alternativas para chamar a atenção da multidão para discutir os elementos ainda não cobertos devem ser propostas, com o objetivo de aumentar a cobertura dos elementos de API e, consequentemente, viabilizar a documentação de API pela multidão.

Agradecimentos

Este trabalho foi parcialmente financiado pela FAPEMIG, CAPES e CNPq.

Referências

Jiau, H. C. e Yang, F.-P. (2012). Facing up to the Inequality of Crowdsourced API Documentation. *ACM SIGSOFT Software Engineering Notes*, 37(1):1–9.

Nasehi, S. M., Sillito, J., Maurer, F., e Burns, C. (2012). What Makes a Good Code Example? A Study of Programming Q&A in StackOverflow. In *Proc. of the ICSM'2012*, pages 25–34.

Parnin, C. e Treude, C. (2011). Measuring API Documentation on the Web. In *Proc. of the Web2SE'2011*, pages 25–30.

Parnin, C., Treude, C., Grammel, L., e Storey, M.-A. (2012). Crowd Documentation: Exploring the Coverage and the Dynamics of API Discussions on Stack Overflow. Technical Report GIT-CS-12-05, Georgia Institute of Technology.

Robillard, M. P. (2009). What Makes APIs Hard to Learn? Answers from Developers. *IEEE Software*, 26(6):27–34.

Souza, L. B., Campos, E. C., e Maia, M. A. (2014a). On the Extraction of Cookbooks for APIs from the Crowd Knowledge. In *Proc. of the SBES'2014*, pages 21–30.

Souza, L. B. L. d., Campos, E. C., e Maia, M. A. (2014b). Ranking Crowd Knowledge to Assist Software Development. In *Proc. of the ICPC'2014*, pages 72–82.

Treude, C., Barzilay, O., e Storey, M.-A. (2011). How Do Programmers Ask and Answer Questions on the Web? In *Proc. of the ICSE'2011*, pages 804–807.