UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CROWD COOKBOOKS: USANDO CONHECIMENTO DE MULTIDÃO A PARTIR DE SÍTIOS DE PERGUNTAS E RESPOSTAS PARA DOCUMENTAÇÃO DE APIS

LUCAS BATISTA LEITE DE SOUZA

Uberlândia - Minas Gerais
2014

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS BATISTA LEITE DE SOUZA

CROWD COOKBOOKS: USANDO CONHECIMENTO DE MULTIDÃO A PARTIR DE SÍTIOS DE PERGUNTAS E RESPOSTAS PARA DOCUMENTAÇÃO DE APIS

Dissertação de Mestrado apresentada à Faculdade de Ciência da Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Engenharia de Software.

Orientador:
Prof. Dr. Marcelo de Almeida Maia

Uberlândia, Minas Gerais
2014

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Ciência da Computação a aceitação da dissertação intitulada "***Crowd Cookbooks*: Usando Conhecimento de Multidão a partir de Sítios de Perguntas e Respostas para Documentação de APIs**" por **Lucas Batista Leite de Souza** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 23 de Julho de 2014

Orientador: _____
Prof. Dr. Marcelo de Almeida Maia
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Prof. Dr. Autran Macêdo
Universidade Federal de Uberlândia

_____
Prof. Dr. Marco Tulio Valente
Universidade Federal de Minas Gerais

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: Julho de 2014

Autor:	**Lucas Batista Leite de Souza**
Título:	***Crowd Cookbooks*: Usando Conhecimento de Multidão a partir de Sítios de Perguntas e Respostas para Documentação de APIs**
Faculdade:	**Faculdade de Ciência da Computação**
Grau:	**Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

_____

Autor

Dedicatória

Aos meus pais Liliane e Márcio e à minha irmã Daniela.

Agradecimentos

Agradeço primeiramente a Deus por ter me concedido inteligência, dedicação e coragem para desenvolver este trabalho. Em segundo lugar ao apoio dos meus familiares que sempre me incentivaram nos momentos mais difíceis.

Também agradeço ao prof. Marcelo Maia pela dedicação e amizade apresentadas na figura de orientador. Ingressamos em uma linha de pesquisa completamente nova para ambos, mas graças a Deus e ao trabalho árduo já estamos colhendo resultados positivos.

Agradeço a todos os docentes que foram meus professores nessa jornada: Sandra de Amo, Gina Oliveira e Michel Soares. O conhecimento obtidos com a ajuda deles foi fundamental na minha formação.

Agradeço a todos os discentes que foram meus colegas ao longo dessa jornada. Em especial: Eduardo Campos, Allan Kardec Soares, Jaqueline Pappini, Juliete Ramos, Cleiane Oliveira e Eldane Vieira. O companherismo e amizade deles foram fundamentais para eu conseguir chegar até aqui.

Agradeço a todos colegas e amigos que perderam algumas horas para me ajudar na validação do meu trabalho: Bruno Moraes, Carlos Sabino, Claudio Sousa, Dennis Catrário, Diego Silva, Diogo Nunes, Eduardo Vilarinho, Fernanda Madeiral, Guilherme Constantin, Juliete Ramos, Liliane Nascimento, Marcos Martins e Renato Juliano. Sem a ajuda deles seria impossível eu chegar até aqui.

If we understand the revolutionary transformations caused by new media, we can anticipate and control them; but if we continue in our self-induced subliminal trance, we will be their slaves.
(Marshall McLuhan)

Resumo

Desenvolvedores de elementos reusáveis de software, como as bibliotecas, em geral têm a responsabilidade de disponibilizar documentação abrangente e de alta qualidade para permitir o reuso efetivo desses elementos. O reuso efetivo de bibliotecas depende da qualidade da documentação da API (Interface para Programação de Aplicativos). Bibliotecas bem estabelecidas tipicamente têm documentação abrangente, por exemplo em Javadocs. Porém, essa documentação geralmente carece de exemplos e explicações, o que pode dificultar o reuso efetivo da biblioteca.

Stackoverflow.com (SO) é um serviço de perguntas e respostas (Q&A) direcionado a questões relacionadas ao desenvolvimento de software. No SO, um desenvolvedor pode postar perguntas relacionadas a um tópico de programação e outros membros do site podem disponibilizar respostas para ajudá-lo a resolver o problema que ele tem em mãos. Apesar da utilização crescente do SO pela comunidade de desenvolvimento de software, a informação relação a um biblioteca está espalhada ao longo do site. Assim, o SO ainda carece de uma organização do *crowd knowledge* nele contido.

Nessa dissertação, será apresentada uma abordagem semi-automatizada que organiza a informação disponível no SO para a construção de um tipo de documentação para APIs, conhecido por *cookbooks* (livros orientados a receitas). Os *cookbooks* produzidos pela abordagem proposta são chamados *crowd cookbooks*.

Para avaliar a abordagem proposta foram gerados *cookbooks* para três APIs amplamente utilizadas pela comunidade de desenvolvimento de software: SWT, LINQ e QT. Foram identificadas características desejáveis de *cookbooks* e realizado um estudo com sujeitos humanos para entender em que grau os *cookbooks* construídos atendem a estas características. Por meio estudo também foi possível compreender melhor os perfis de uso dos *cookbooks* mais apropriados em relação ao aprendizado de APIs. Os resultados mostraram que os *cookbooks* construídos pela estratégia proposta, em geral, atendem às características identificadas. Além disso, a maior parte dos sujeitos humanos considerou que *cookbooks* não possuem um formato adequado ao aprendizado de APIs.

Palavras chave: sítios de perguntas e respostas, conhecimento da multidão, interface para programação de aplicativos, documentação de software.

Abstract

Developers of reusable software elements, such as libraries, usually have the responsibility to provide comprehensive and high quality documentation to enable effective reuse of those elements. The effective reuse of libraries depends upon the quality of the API (Application Program Interface) documentation. Well established libraries typically have comprehensive API documentation, for example in Javadocs. However, they also typically lack examples and explanations, which may difficult the effective reuse of the library.

StackOverflow.com (SO) is a Question and Answer service directed to issues related to software development. In SO, a developer can post questions related to a programming topic and other members of the site can provide answers to help him/her solve the problem he/she has at hand. Despite of the increasing use of SO by the software development community, the information related to a particular library is spread along the website. Thus, SO still lacks an organization of its crowd knowledge.

In this dissertation, we present a semi-automatic approach that organizes the information available on SO in order to build a kind of documentation for APIs, called cookbooks (recipe-oriented books). The cookbooks generated by the approach are called *crowd cookbooks*.

In order to evaluate the proposed approach, cookbooks were generated for three APIs widely used by the software development community: SWT,LINQ and QT. Desired features that cookbooks must meet were identified and a study with human subjects was conducted to assess to what extent the generated cookbook meet those features. Through the study it was also possible to identify what is the perceived usefulness by the subjects in relation to the use of cookbooks in APIs learning. The results showed that the cookbooks built using the proposed strategy, in general, meet the identified features. Furthermore, most human subjects considered that cookbooks do not have an appropriate format to the learning of APIs.

Keywords: questions and answers sites, crowd knowledge, application programming interface, software documentation.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Nas duas últimas décadas, concomitantemente com a emergência da Web 2.0, uma nova cultura e filosofia emergiram e estão mudando as características do desenvolvimento de software. Esta mudança é o resultado de uma estrutura acessível de mídias sociais (*wikis*, *blogs*, *fóruns*, sites de perguntas e respostas. etc) [49]. Da mesma forma que o desenvolvimento de código-livre tem mudado o processo tradicional de desenvolvimento de software [43], essas novas formas de contribuição e colaboração têm o potencial de redefinir como desenvolvedores aprendem, preservam e compartilham conhecimento sobre desenvolvimento de software.

Um exemplo importante de mídia social é o Stack Overflow[1] (SO) que é um *website* de Perguntas e Respostas (*Questions and Answers* - Q&A) que permite a troca de conhecimento entre desenvolvedores, mitigando as dificuldades envolvidas em utilizar código-fonte da Internet. O *design* do SO nutre uma comunidade de desenvolvedores e permite atividades de Engenharia de Software *crowdsourced*, desde documentação, até a disponibilização trechos de código úteis e de alta qualidade [5]. O termo *Crowdsourcing*[2] é usado para se referir ao processo de obter serviços necessários, ideias ou conteúdo, solicitando contribuições de um grande grupo de pessoas e, especialmente, a partir de uma comunidade *online*, ao invés de empregados ou fornecedores tradicionais. Segundo Barzilay et al. [5], o conjunto de informações disponíveis nesse tipo de serviços de mídia social é chamado de "conhecimento da multidão" (*crowd knowledge*) e geralmente se torna um substituto para documentação oficial de software.

Com o objetivo de descobrir o que está por trás do sucesso do SO, Mamykina et al. [40], conduziram um estudo estatístico em todo o corpo de conhecimento do *site*. Os achados do estudo mostraram que a maioria das perguntas recebe uma ou mais respostas (mais de 90% muito rapidamente - com a mediana do tempo de resposta igual a 11 minutos). Treude et al. [64] descobriram que o SO é particularmente efetivo para revisão de código-fonte, perguntas conceituais e perguntas de novatos.

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

[2] http://www.merriam-webster.com/dictionary/crowdsourcing <Acesso em 18/06/2014>

Parnin et al. [49] desenvolveram um estudo com o objetivo de compreender o potencial que o *crowd knowledge* possui em relação a documentação de APIs (*Application Program Interfaces*). Os autores verificaram que no SO existe um alto percentual de métodos e classes abordados das APIs estudadas (e.g., 87% das classes da API Android são referenciadas em perguntas no SO). Outro resultado encontrado por Parnin et al. é que a maior parte das respostas no SO são disponibilizadas por *experts* em programação de software.

Sabe-se que a documentação de software é algo importante para o reuso de software. De acordo com Parnas (citado em [8] p. 224) "*Reuso é algo que é muito mais fácil dizer do que fazer. Fazê-lo requer bom design e documentação de qualidade. Mesmo quando nós vemos bom design, o que é não é frequente, nós não vemos os componentes sendo usados sem boa documentação*".

Tradicionalmente, muitos tipos de documentação de software, como documentação de APIs, são gerados por poucas pessoas e têm um público alvo muito maior. A documentação resultante, quando existe, geralmente tem baixa qualidade e carece de exemplos e explicações [49]. Um estudo desenvolvido por Robillard [53], mostrou que desenvolvedores consideram o uso de exemplos como um dos principais recursos ao aprendizado de APIs. De acordo com ČubraniĆ et al. [67], estudar a partir de exemplos e abstraí-los em um novo contexto é uma maneira comum de aprendizado tanto para programadores iniciantes quanto experientes.

O fato de o *crowd knowledge* do SO possuir um grande volume de informações sobre APIs, com soluções disponibilizadas em sua maioria por *experts*, somado ao fato de que a documentação de APIs, quando existe, geralmente carece de exemplos e explicações, servem de motivação à tentativa de construir documentação para APIs a partir das informações disponíveis no SO.

Nesta dissertação será apresentada uma abordagem semi-automatizada que faz uso das informações do SO para a construção de um tipo de documentação para APIs conhecido por *cookbooks*. Esse tipo de documentação é organizado em capítulos e cada capítulo agrupa um conjunto de receitas que tratam do mesmo assunto. Cada receita apresenta um problema e instruções sobre como resolver este problema, muitas vezes com explicações em linguagem natural e trechos de código-fonte. Os *cookbooks* construídos pela estratégia proposta são chamados de *crowd cookbooks*. *Cookbook* é um tipo de documentação orientado à exploração (*browsing*), ao contrário dos mecanismos tradicionais de busca (*searching*). Mais detalhes sobre a natureza exploratória dos *cookbooks* são discutidos no Capítulo 3.

Outra motivação para o trabalho apresentado nesta dissertação é que apesar de o SO possuir um grande volume de informações sobre APIs, pode-se dizer que falta uma certa organização/estruturação das informações lá disponíveis. Um fato que demonstra isso é o número de perguntas no SO relativos à um particular tópico (e.g., existem mais de 41.000 perguntas relacionadas à bibliotecas Swing) que estão "espalhadas" pelo *site*

(dado referente à maio de 2014). Apesar de que no SO o usuário pode navegar por todas as *threads* que possuem uma particular *tag*, não existe uma estruturação que o permita verificar quais são os principais temas existentes para aquela *tag*.

O objetivo da estratégia de construção de *cookbooks* apresentada nesta dissertação é organizar/estruturar o conteúdo do SO sobre APIs, pois os capítulos de um *cookbook* sumarizam os assuntos existentes para a API alvo do *cookbook*.

As contribuições do trabalho descrito nesta dissertação são:

- A descrição de um algoritmo para construção de *crowd cookbooks* sobre APIs existentes no SO;
- A aplicação do algoritmo proposto para a construção de *cookbooks* para três APIs comumente utilizadas pela comunidade de desenvolvimento de software, relacionadas a diferentes linguagens de programação: SWT (Java), LINQ (linguagens .NET) e QT (C++).
- A identificação de propriedades desejáveis de *cookbooks* e de critérios que permitem mensurá-las;
- O desenvolvimento de um estudo com sujeitos humanos para avaliar os *cookbooks* gerados no que diz respeito aos critérios definidos e para entender o papel que *cookbooks* podem desempenhar no aprendizado de APIs.

O restante desta dissertação está organizada da seguinte maneira. O Capítulo 2 apresenta algumas definições importantes ao entendimento da estratégia utilizada para construção de *crowd cookbooks*. O Capítulo 3 apresenta a abordagem para construção de *crowd cookbooks*. A metodologia proposta para avaliação da estratégia de construção de *crowd cookbooks* está na Capítulo 4. Os resultados são apresentados e discutidos no Capítulo 5. Os trabalhos relacionados são revistos no Capítulo6. As conclusões e trabalhos futuros são apresentados no Capítulo 7.

Capítulo 2

Conceitos Introdutórios

Neste capítulo apresentaremos algumas definições importantes que são necessárias para o entendimento da abordagem de construção de *cookbooks* apresentada nesta dissertação. A Seção 2.1 apresenta o conceito de mídia social. A Seção 2.2 apresenta o Stack Overlflow, que é a fonte de dados utilizada para construção dos *crowd coookboks*. A Seção 2.3 apresenta os diferentes tipos de perguntas que são feitas no SO, uma vez que somente um desses tipos interessa aos *crowd coookboks*. Por fim, a Seção 2.4 apresenta o conceito de *Application Program Interfaces*, pois os *cookbooks* construídos pela estratégia proposta visam a documentação de APIs.

2.1 Mídia Social

O trabalho apresentado nesta dissertação faz uso do SO como fonte de informação usada para construção dos *cookbooks*. SO é um tipo de mídia social. Dessa forma, é importante definir o que é mídia social.

Barzilay et al. apresentaram em [5], algumas definições sobre mídia social. As definições apresentadas nesta seção foram retiradas deste trabalho. "*Mídia social é um termo abrangente que define as várias atividades que integram tecnologia e interação social, habilitada pelos avanços recentes nas tecnologias da Web 2.0*" [5]. A organização W3C define mídia social como "*Tecnologias e práticas online que pessoas usam para compartilhar opiniões, insights, experiências e perspectivas*" [1].

Segundo Barzilay et al., "*Mídia social provê recomendações úteis para muitas áreas da nossa vida. Por exemplo, quando se está em dúvida sobre quais filmes assistir, alguém pode usar recomendações de seu círculo social (e.g., amigo do Facebook), ou utilizar a sabedoria da multidão, considerando, por exemplo, as notas no imdb.com. Isso é parte de tendência mais geral na qual recomendações via mídia socais (e.g., Facebook) começou a substituir a busca (e.g., Google Search).*".

[1] http://www.w3.org/egov/wiki/Glossary

Ainda segundo Barzilay et al., a mídia social se mostrou benéfica em várias áreas da Engenharia de Software, como priorização de características [3], análise de riscos [61], filtragem colaborativa [25], gerenciamento de conhecimento [26] e documentação [7]. A mídia social está mudando a maneira pela qual desenvolvedores se comunicam e coordenam e como eles produzem e consomem documentação [65].

2.2 Stack Overflow

Stack Overflow é um *website* de Perguntas e Respostas (Q&A) no qual desenvolvedores postam perguntas e outros desenvolvedores fornecem respostas. O questionador (i.e., a pessoa que posta uma pergunta) pode adicionar até cinco rótulos (*tags*) para ajudar os outros (e.g., os potenciais respondentes) descobrir sobre o que a pergunta trata. O questionador pode selecionar uma resposta como a mais útil (a chamada resposta aceita). Os membros do site podem votar em perguntas e respostas. Os votos negativos e positivos (chamados de *upvotes* e *downvotes* respectivamente) mostram o quão útil a pergunta/resposta foi para o público. A diferença entre o número de *upvotes* e *downvotes* determina a nota (*score*) de uma pergunta/resposta. Cada membro do site tem uma reputação determinada por um *score de reputação*. À medida que eles participam em diferentes atividades do SO, como postando perguntas ou respostas, votando nas mesmas, postando comentários, etc., o *score de reputação* aumenta e valores maiores de reputação indicam mais capacidades. Por exemplo, usuários com uma reputação acima de um certo *threshold* podem editar perguntas/respostas ou fechar uma *thread* de Q&A. A natureza interativa do SO torna possível para ambos, questionadores e respondentes, clarear a imprecisão em uma pergunta/resposta. Até mesmo outros membros do site podem editar peguntas/respostas, caso achem necessário. *Badges* (i.e., insígnias) são dadas aos usuários pelas suas contribuições uma vez que eles atingem um certo *threshold*.

Mamykina et al. [40], conduziram um estudo estatístico em todo o corpo de conhecimento do SO para descobrir o que está por trás do sucesso deste site. Segundo Mamykina et al., "*Dentro de dois anos, o SO se tornou uma das mais visíveis vias para compartilhamento de conhecimento sobre desenvolvimento de software. Com aproximadamente 300.000 usuários registrados e mais de 7 milhões de visitas mensais (em agosto de 2010), SO tem uma taxa de resposta acima de 90% e uma mediana do tempo de resposta igual a 11 minutos. Usuários reportam que o site substituiu a busca na web e os fóruns como a fonte primária para resolução de problemas; outros consideram o seu portfólio de respostas no SO como um componente precioso de seus currículos profissionais.*". As principais razões encontradas pelo estudo de Mamykina et al. para explicar o sucesso do *site* incluem: 1) o forte envolvimento dos fundadores do *site* com a comunidade; 2) uma abordagem altamente iterativa e responsiva para projetar o site (e.g., com frequência são lançadas novas versões do site, para melhor atender às expectativas dos usuários); 3) um sistema

de incentivos (e.g., reputação) que promove comportamentos desejáveis dos usuários.

A Figura 2.1 mostra os principais elementos de uma *thread* de discussão no SO. Nesta dissertação o termo *thread* será usado para referir ao conjunto formado por uma pergunta e todas respostas a esta pergunta. O termo *post* será usado para referir a uma pergunta ou resposta no SO. Observe que no corpo de um *post* pode haver tanto texto em linguagem natural (inglês) como trechos de código-fonte.

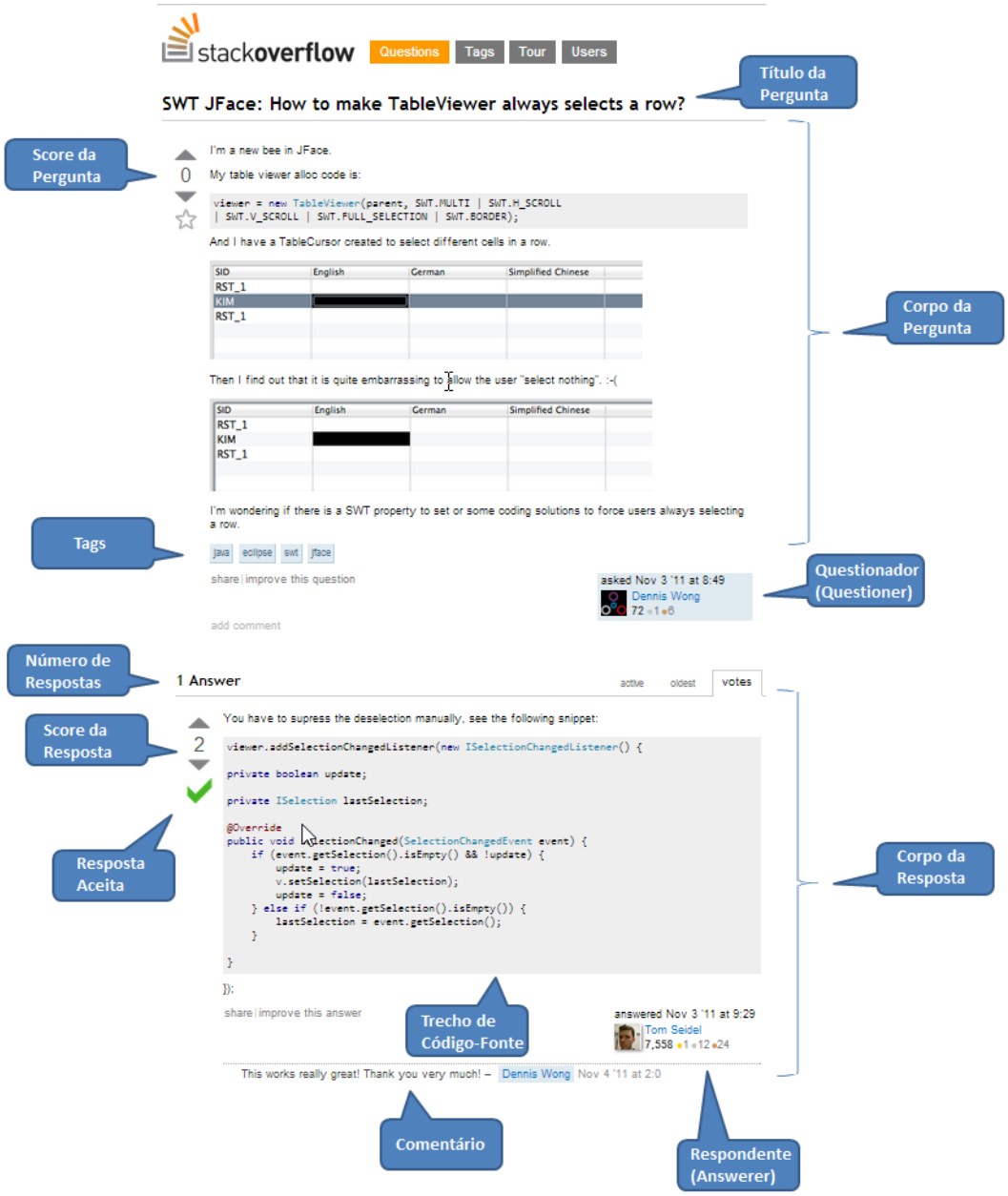


Figura 2.1: Principais elementos de uma *thread* no SO

Como o estudo de Mamykina et al. foi feito em agosto de 2010 e o SO tem uma rápida taxa de crescimento, as estatísticas apresentadas no trabalho foram recalculadas (em março de 2013) e são mostradas a seguir:

- Número de usuários cadastrados: 1.785.608;

- Número de perguntas postadas: 4.592.961;
- Número de respostas postadas: 8.639.860;
- Número de comentários postados a perguntas ou respostas: 8.639.860;
- Média do score de perguntas: 1,5464;
- Desvio padrão do score de perguntas: 1,5464;
- Média do score de respostas: 1,9769;
- Desvio padrão do score de respostas: 8,7068;
- Número de perguntas com presença de código-fonte: 2.582.653 (56,23%);
- Número de respostas com presença de código-fonte: 3.998.375 (46,27%);
- Número de perguntas com pelo menos uma resposta: 4.153.159 (90,42%);
- Número de perguntas com resposta aceita: 2.808.543 (61,14%);
- Mediana do tempo decorrido para cada pergunta receber a primeira resposta: 16,18 minutos;
- Mediana do tempo decorrido para cada pergunta receber a resposta que será escolhida como resposta aceita: 24,45 minutos;
- Número de perguntas que receberam a primeira resposta em no máximo 1 hora: 2829047 (61,59%);
- Número de perguntas que receberam a primeira resposta em no máximo 1 dia: 3674488 (80%);
- Número de perguntas que receberam a resposta que viria a ser escolhida como aceita, em no máximo 1 hora: 1763981 (62,80%);
- Número de perguntas que receberam a resposta que viria a ser escolhida como aceita, em no máximo 1 dia (24 horas): 2435612 (86,72%);
- Número médio de respostas para as perguntas que receberem pelo menos 1 resposta: 2,08;
- Score médio das respostas que são escolhidas como aceitas: 3,14.

Os dados utilizados nesta dissertação para construir os cookooks utilizando a estratégia proposta são provenientes de um *dump* que o time do SO disponibiliza a cada três meses. Esse *dump* contém todo o conteúdo do SO, incluindo *posts*, usuários, votos, etc., desde a data de criação do SO (em 2008) até o momento de geração do *dump*. A versão utilizada foi a de setembro de 2013[2]. Os *dumps* obtidos foram importados em um banco de dado relacional do Mysql Server. Optou-se em construir um banco de dados relacional ao

[2]http://meta.stackexchange.com/questions/198915/is-there-a-direct-download-link-with-a-raw-data-dump-of-stack-overflow-not-a-t/199303#199303

invés de trabalhar diretamente como o *dump* (que internamente armazena os dados no formato XML), pelo fato de que fazer consultas em um BD relacional é mais rápido do que consultar arquivos XML. O time do Stack Overflow também disponibiliza uma API REST[3] para acesso aos dados do SO por aplicações de terceiros. Entretanto, como este serviço possui algumas limitações (e.g., existe um número máximo de requisições que podem ser feitas por dia), optou-se por utilizar o *dump*.

2.3 Tipos de Perguntas no SO

No SO, usuários perguntam muitos tipos de perguntas. De acordo com Nasehi et al. [46], "*Perguntas no SO podem ser descritas em duas dimensões diferentes. A primeira dimensão diz respeito ao tópico da pergunta: a principal tecnologia ou construção que a pergunta gira em torno e geralmente pode ser identificada a partir das tags que o questionador adiciona à pergunta para ajudar os outros (e.g., respondentes potenciais) a descobrirem sobre o que pergunta trata.*". Por exemplo, a pergunta mostrada na Figura 2.1 é sobre a biblioteca SWT, pois "swt" é uma das *tags* presentes na pergunta. Ainda de acordo com Nasehi et al., "*perguntas no SO podem também ser classificadas em uma segunda dimensão que é sobre as principais preocupações do questionador e o que ele quer resolver*". Nesta dimensão, Nasehi et al. definiram um conjunto de quatro categorias:

- *Debug/Corrective*: Lida com problemas no código em desenvolvimento, como erros em tempo de execução, comportamento inesperado, erros de compilação. Essa categoria também pode ser sobre um código que está funcionando mas não está satisfatório devido a seu *design* ou estrutura e por isso o usuário está procurando por um melhor;
- *Need-To-Know*: Perguntas que dizem respeito a possibilidade ou disponibilidade de fazer algo. Essas perguntas geralmente mostram a falta de conhecimento ou incerteza sobre alguns aspectos da tecnologia (e.g., a presença de uma funcionalidade em uma API ou linguagem);
- *How-To-Do-It*: Disponibilizam um cenário e pedem por instruções sobre como implementá-lo (algumas vezes utilizando uma tecnologia ou API);
- *Seeking-Different-Solution*: O questionador tem um código que está funcionando mas procura por uma abordagem diferente para fazer a tarefa.

Treude et al. [64] definiram uma taxonomia diferente, composta por dez categorias: *how-to*, *discrepancy*, *environment*, *error*, *decision help*, *conceptual*, *review*, *non-functional*, *novice* e *noise*. A categoria *how-to* é definida de maneira semelhante à *How-To-Do-It*: "*Perguntas que pedem por instruções, e.g., como cortar uma imagem*". Esta categoria

[3] http://api.stackexchange.com/

se mostrou a mais comum na amostra analisada por Treude et al., constituindo 39,22% das perguntas analisadas. Essa categoria (*How-To-Do-It* ou *how-to*) é muito próxima do conceito de receitas de um *cookbook*, pois assim como as perguntas *How-To-Do-It*, as receitas de *cookbook* também contém um cenário e instruções sobre como implementá-lo com o uso de uma tecnologia ou API. Por essa razão, na abordagem apresentada nesta dissertação, serão consideradas apenas perguntas que se enquadram nesta categoria. No Capítulo 3 será apresenta uma estratégia baseada em regras que permite a identificação deste tipo de pergunta.

2.4 *Application Program Interfaces*

Uma biblioteca pode ser definida como a implementação de uma funcionalidade com a intenção de ser reusada por terceiros. Nos casos das linguagens de programação orientadas a objeto, como Java e C++, uma biblioteca consiste de uma coleção coerente de classes. Em geral, uma biblioteca tem duas partes: a *interface pública* e a implementação privada. A primeira parte contém elementos de software (por exemplo, classes e métodos) que outros desenvolvedores podem usar em seu próprio código-fonte. Esta parte é chamada de Interface de Programação de Aplicativos (*Application Programming Interface* ou API) [44]. A segunda parte implementa a funcionalidade da biblioteca, mas geralmente não é exposta aos usuários da biblioteca. Os desenvolvedores destes aplicativos podem usar os serviços de uma API através, por exemplo, da inclusão de chamadas a métodos definidos naquela API em seus aplicativos.

APIs são um tipo de abstração que possibilita o reuso caixa-preta (*black-box reuse*), pois permitem que uma biblioteca seja usada sem ser necessário conhecer os seus detalhes internos de implementação.

Como a API é a parte que é exposta aos usuários de uma biblioteca, a documentação de uma biblioteca deve documentar a sua API. Por essa razão, a abordagem apresentada nesta dissertação é voltada à construção de um tipo de documentação (*cookbooks*) para APIs.

Capítulo 3

Abordagem Semi-Automatizada Para Construção de *Crowd Cookbooks*

Segundo Sametinger [55], os exemplos mais proeminentes do reuso de software atual incluem algoritmos, bibliotecas, *framework* de aplicações e padrões de projetos. "*Um aspecto importante para o reuso de bibliotecas é que elas devem conter uma interface. Deve haver algum tipo de abstração e o reuso deve ser possível sem conhecer os detalhes internos da biblioteca. Esse tipo de reuso é chamado de reuso caixa-preta (black-box reuse). De maneira a aumentar a produtividade de software, o reuso caixa-preta deve ser utilizado.*". Ainda de acordo com Sametinger, documentação é necessária para o reuso.

Tradicionalmente, muitos tipos de documentação de software, como documentação de APIs, são gerados por poucas pessoas e têm um público alvo muito maior. A documentação resultante, quando existe, geralmente tem baixa qualidade e carece de exemplos e explicações [49]. Um estudo desenvolvido Robillard [53], mostrou que desenvolvedores consideram o uso de exemplos como um dos principais recursos ao aprendizado de APIs. De acordo com ČubraniĆ [67], estudar a partir de exemplos e abstraí-los em um programa em um novo contexto é uma maneira comum de aprendizado tanto para programadores iniciantes quanto experientes.

Como os desenvolvedores geralmente não têm todas as informações que eles precisam para completar uma tarefa particular de programação, mesmo consultando a documentação oficial da(s) biblioteca(s) que ele está utilizando, a alternativa mais próxima é pesquisa na Web [5]. Um estudo de Hoffmann [28] mostrou que quando desenvolvedores pesquisam na Web, a maior parte de suas consultas é sobre APIs. Para fazer uma busca sobre uma API, um desenvolvedor pode, por exemplo, utilizar o Google, pesquisar no próprio mecanismo de busca existente no SO ou utilizar abordagens que recomendam conteúdo utilizando dados do SO [19, 51]. Esse processo de consulta é conhecido como *searching*.

Segundo Olston e Chi [47], "*Searching é o processo de entrar com uma consulta (geralmente uma lista de palavras-chave) em uma máquina de busca, que produz uma lista ranqueada de páginas que casam com a consulta.*".

Ortogonalmente ao mecanismo de *Searching*, está a estratégia de *browsing*. "*Browsing é uma estratégia exploratória de busca de informação que depende da serendipidade e é adequada especialmente para problemas mal-definidos e para explorar novos domínios*" [41]. *Browsing* é caracterizado pela ausência de planejamento e é geralmente usado como uma alternativa à estratégia de busca baseada em palavras-chave. "*Em essência, o processo de browsing explora tanto a organização do espaço de informação, como o seu conteúdo*" [13].

Como *browsing* é frequentemente usado em espaços de informação novos ou relativamente novos (não explorados), usuários tipicamente dependem de modelos mentais de organização da informação pré-existentes quando eles exploram o espaço. Modelos mentais são definidos como "*representações cognitivas de um problema (ou informação), situação ou sistema*" [13]. Elas ajudam o usuário a representar o conteúdo, estrutura e relação das informações no espaço de informação. Isso ajuda o usuário a entender a organização do espaço e responder às condições que ocorrem durante a navegação [13]. Por exemplo, no caso de um livro, esse modelo mental pode estar presente na forma do sumário.

Apesar de o processo de *browsing* e *searching* terem desvantagens, ambos persistem devido às sua vantagens complementares. *Searching* é popular devido à sua habilidade de identificar rapidamente itens contendo uma informação específica. Por outro lado, *browsing* é útil nos casos em que palavras-chave de busca apropriadas são inexistentes ou não disponíveis ao usuário devido a várias possíveis razões (e.g., por se tratar de um domínio de aplicação novo para o usuário, o mesmo pode não ser capaz de derivar palavras-chave para uma consulta) [47].

Cookbook é um tipo de documentação que faz uso da estratégia de *browsing*. Em um *cookbook*, cada receita é um item que contém um problema computacional e instruções sobre como resolvê-lo. Geralmente as receitas são agrupados em capítulos, cada um dos quais agrupa receitas de uma área de aplicação específica. Por exemplo, em um *cookbook* sobre a linguagem Java, poderia existir um capítulo sobre manipulação de Strings e outro relacionado com a manipulação de arquivos XML. Cada capítulo possui um título que descreve o assunto referente às receitas nele contidas. A organização do *cookbook* em capítulos é o modelo mental que permite a exploração do mesmo. Cada capítulo pode ser considerado uma categoria (assunto). Ao ler os títulos dos capítulos, o usuário obtém uma ideia geral do que o *cookbook* trata, bem como do assunto individual de cada tópico. Os *cookbooks* possuem apenas dois níveis hierárquicos (capítulos e receitas), o que faz com o que a exploração dos mesmos seja um processo simples.

Em um *cookbook*, não existe uma ordem cronológica forte entre as receitas e os capítulos, ao contrário de outros tipos de documentação (e.g., tutoriais), que exigem uma leitura sequencial (passo-a-passo). A maioria dos leitores experimenta as receitas de forma aleatória ou com base em seu interesse do momento. Em outras palavras, os *cookbooks* exploram a estratégia de *browsing*.

O termo *cookbook* foi originalmente utilizado na culinária e foi trazido para o contexto de computação, onde é usado para se referir a uma enumeração de receitas *How-To-Do-It*. O uso deste termo em Ciência da Computação não é algo novo. Na área de software, Donald Knuth aplicou a analogia de "*cookbook*" no prefácio de seu livro "The Art of Computer Programming" [32], primeiramente publicado em 1968. Na área de hardware, Don Lancaster escreveu em 1974 "The TTL *Cookbook*" [37] (TTL era o bloco de construção de pequena escala de circuitos eletrônicos da época).

Existem uma grande quantidade de *cookbooks* disponíveis à venda. Em uma pesquisa no site de comércio eletrônico Amazon [1], foram encontrados 577 resultados para o termo "*cookbook*" na área de "Computer & Technology -> Programming". Dentre os resultados existem *cookbooks* para linguagens de programação, *frameworks*, bibliotecas, plataformas, etc.

Neste capítulo será apresentada a estratégia desenvolvida para geração de *cookbooks* a partir de dados do SO. Os principais passos são ilustrados na Figura 3.1. Os dados utilizados para construção dos *cookbooks* são provenientes do banco de dados (Figura 3.1.a) construído a partir do *dump* disponibilizado pela equipe do SO (versão de setembro de 2013). Suponha que esteja sendo gerado um *cookbook* para uma API X qualquer. O primeiro passo para a construção de um *cookbook* para a API X é selecionar todas as *threads* que dizem respeito a esta API. Para isso, utiliza-se a informação presente nas *tags* das perguntas do SO. Portanto, neste caso deve-se selecionar todas *threads* com a *tag* "X" (Figura 3.1.b). O próximo passo é identificar dentre todas as *threads* do SO quais delas possuem pergunta do tipo *How-To-Do-It*, pois somente esse tipo de *thread* será considerada para construção dos *cookbooks* (Figura 3.1.c). Depois disso, as *threads* devem ser submetidas a um pré-processamento (Figura 3.1.d) antes da aplicação da técnica de modelagem de tópicos (LDA) (Figura 3.1.e) que é responsável por encontrar os potenciais capítulos do *cookbook* sendo construído. Depois da aplicação do LDA, o algoritmo de construção dos *cookbooks* (Figura 3.1.f) utiliza as informações geradas pela LDA para de fato construir o *cookbook*, i.e., determinar quais serão os capítulos e receitas do mesmo.

A seguir será explicado como as perguntas *How-To-Do-It* são identificadas (Seção 3.1); o pré-processamento dos dados conduzido (Seção 3.2) antes da aplicação do técnica de modelagem de tópicos (Seção 3.3) utilizada para descobrir os potenciais capítulos de um *cookbook*; o algoritmo utilizado para construção de *cookbooks* (Seção 3.4); um estudo conduzido com o objetivo de determinar os valor de alguns dos parâmetros utilizados na construção dos *cookbooks* (Seção 3.5). Além disso, foi construída uma aplicação *web* para permitir a visualização dos *crowd cookbooks* (Seção 3.6).

[1] http://www.amazon.com/ <Pesquisa feita em 13/05/2014>

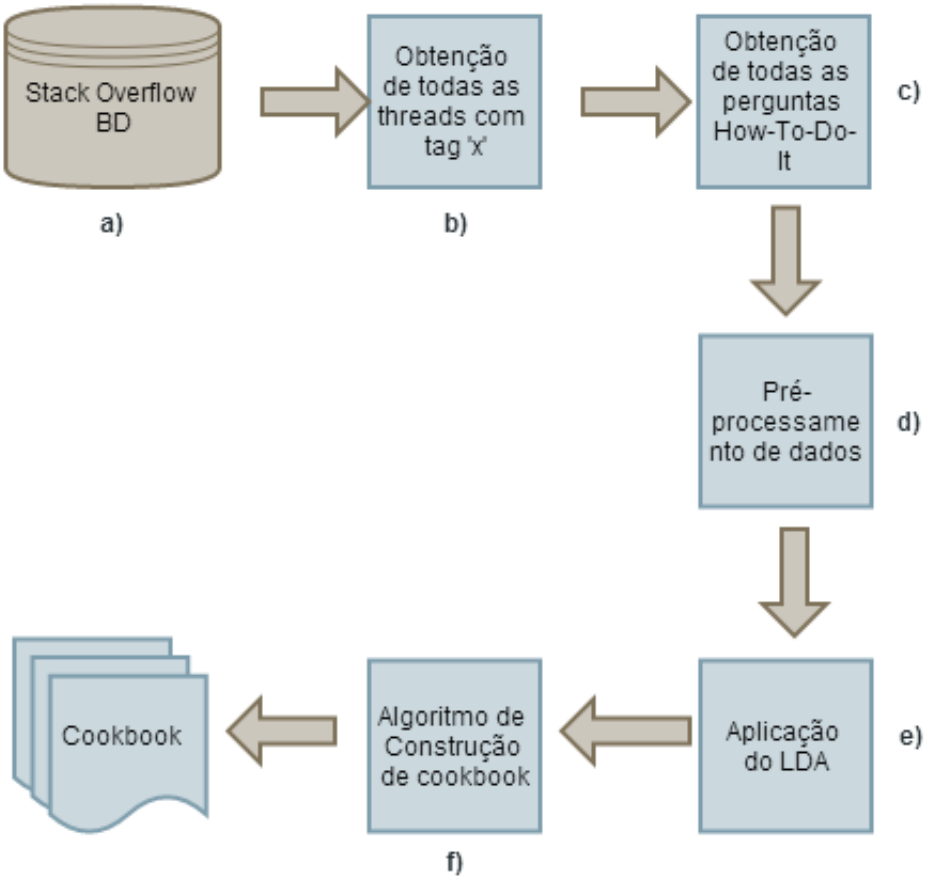


Figura 3.1: Principais passos da estratégia para construção de *cookbooks*

3.1 Identificação de Perguntas *How-To-Do-It*

Foi desenvolvida uma abordagem baseada em regras, que considera os termos presentes no título e corpo das perguntas. Usando esta abordagem, uma pergunta é considerada um *How-To-Do-It* se as três condições a seguir são satisfeitas:

- Ela possui o termo "how" no título ou no texto do corpo;
- Ela não possui no texto em linguagem natural do corpo, termos geralmente relacionados à categoria *Debug/Corrective*: "fail", "problem", "error", "wrong", "fix", "bug", "issue", "solve", "trouble" (ou variações destas palavras, e.g.: "erros", "solved", "fixed");
- Ela não possui a palavra "error" no espaço destinado a trechos de código (caso possua algum).

Essas regras foram definidas a partir da análise de um conjunto de 70 perguntas escolhidas aleatoriamente do banco de dados do SO. A Figura 3.2 mostra um exemplo de pergunta *How-To-Do-It* (observe que a pergunta possui "how" no título). A Figura 3.3 mostra uma pergunta que não foi classificada como *How-To-Do-It* porque possui a palavra

"error" no texto em linguagem natural do corpo e também no espaço destinado a trechos de código (apesar de o conteúdo presente neste campo ser o *log* de uma exceção gerada e não um código-fonte propriamente dito). Na avaliação dos *cookbooks* criados usando a abordagem apresentada neste capítulo, foi definido um critério que permite avaliar se as perguntas selecionadas para fazer parte do *cookbook* são de fato *How-To-Do-It*.

Figura 3.2: Exemplo de pergunta *How-To-Do-It*

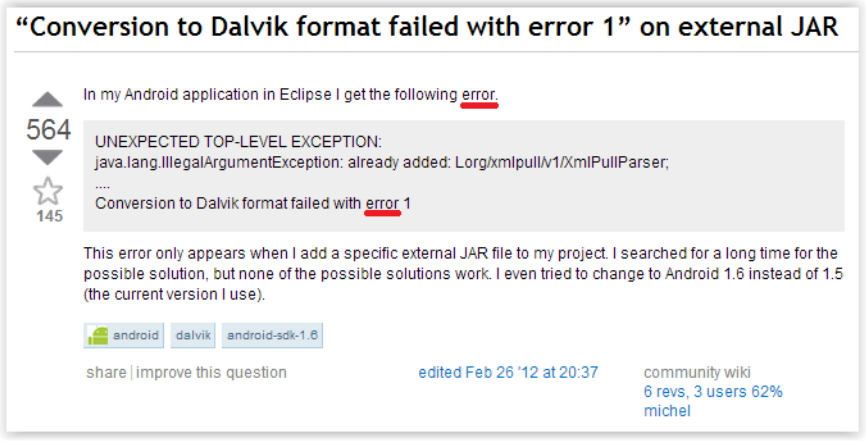


Figura 3.3: Exemplo de pergunta que não é *How-To-Do-It*

3.2 Pré-processamento dos Dados

Os *cookbooks* serão organizados em capítulos, cada um referente a um assunto da API alvo do *cookbook*. Para identificação dos capítulos do *cookbook*, estamos utilizando a técnica de modelagem de tópicos *Latent Dirichlet Allocation* (LDA). LDA é uma técnica de recuperação de informação que encontra automaticamente os temas gerais a partir de um corpo de documentos [6].

Inicialmente, é preciso criar o corpo de documentos que será usado como entrada para o LDA. Para cada *thread* cuja pergunta é um *How-To-Do-It*, é criado um documento contendo o conteúdo textual da pergunta e todas as suas respostas. Esse documento é construído, concatenando-se o título da pergunta, o corpo da pergunta e o corpo de todas as eventuais respostas. Como exemplo, a Figura 3.4.a mostra o conteúdo textual extraído

de uma *thread* do SO (antes do pré-processamento). O conjunto de documentos criado é chamado de *corpo de documentos*.

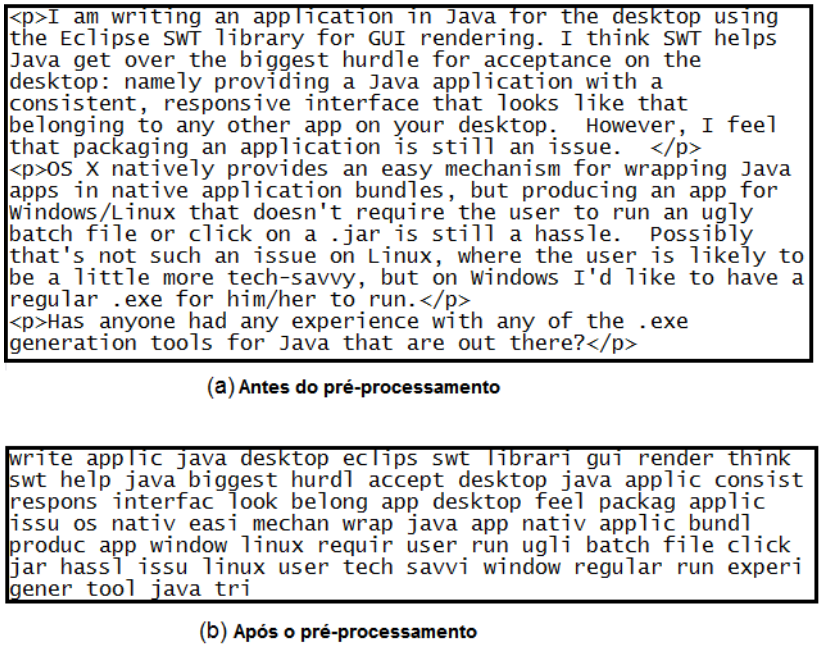


Figura 3.4: Exemplo de documento, antes e após o processamento

Antes de aplicar o LDA é necessário pré-processar o conteúdo dos documentos criados. O pré-processamento conduzido aqui foi baseado no trabalho de Barua et al. [4] e é necessário para eliminar informações que não são úteis à descoberta dos tópicos, sendo feito em quatro passos:

- Os trechos de código-fonte (o conteúdo que está entre *tags* HTML "<code>"), caso existam, são removidos, porque a sintaxe do código-fonte (e.g., laços "while" e "for") introduz ruído para o LDA. Como todos trechos de código-fonte contêm sintaxe e palavras reservadas semelhantes, eles não ajudam o LDA a encontrar tópicos úteis [4,63]. Outro fato que justifica a remoção dos trechos de código, é que a maior parte do código-fonte no SO está presente na forma de pequenos trechos de código, de maneira que não existe contexto suficiente para permitir a extração de conteúdo significativo a partir dos trechos [4, 35]. A remoção de código-fonte ocorre apenas para a aplicação do LDA. Ou seja, os trechos de código-fonte são mostrados nas receitas selecionadas para serem incluídas nos *cookbooks* (precesso exlicado na seção 3.4);
- O próximo passo é remover todas as *tags* HTML (e.g.,
e <a href="..."), porque o nosso foco é analisar conteúdo em linguagem natural (inglês);
- Em seguida, palavras comuns da língua inglesa (chamadas de *stop words*), como "a", "the" e "is", são removidas, pois elas não ajudam na criação de tópicos com temas

significativos. Além das palavras comuns da língua inglesa, foi definido um conjunto de *stop words* específico para o domínio do SO. Termos como "question", "answer", "code", "need", "implement", são comumente usados nos *posts* do SO e não agregam informação útil para a construção dos tópicos, por isso é adequado considerá-los como *stop words*. A lista completa de *stop words* utilizada, pode ser encontrada no Apêndice A.1;

- Por último, é aplicado o algoritmo de Porter Stemming [52], que mapeia as palavras à sua forma base (e.g., "programming" e "programmer" ambos são mapeados para "program"). As formas bases também são chamadas de *stems*.

O Apache Lucene[2] foi utilizado para a remoção de *stop words* e *stemming* e a biblioteca Java HTML Cleaner[3] para remoção de *tags* HTML.

A Figura 3.4.b mostra um documento após ter sido submetido ao pré-processamento. Observe após o pré-processamento, os termos presentes no documento são *stems* ao invés de palavras originais.

3.3 Uso do LDA para Determinação dos Potenciais Capítulos

Latent Dirichlet Allocation (LDA) é uma técnica de recuperação de informação que encontra automaticamente os temas gerais a partir de um corpo de documentos com conteúdo textual, sem a necessidade *tags*, treinamento de dados ou taxonomias pré-definidas [6]. O LDA apenas usa a frequência das palavras em cada documento e frequências da coocorrência das palavras nos documentos para construir um modelo de palavras relacionadas e dessa forma identificar *clusters* de documentos relacionados em um corpo de documentos. Cada *cluster* é chamado de *tópico*. A mineração de tópicos através do uso do LDA permite agrupar os documentos (*threads*) por características comuns e obter informações dos termos que caracterizam os tópicos. Para minerar os tópicos, foi utilizada a biblioteca de código-livre MALLET [42], que é bem estabelecida e já foi usada em diversos outros trabalhos [4,44]. O LDA disponibilizado nesta biblioteca é uma implementação do algoritmo de amostragem Gibbs [22]. Cada tópico minerado pelo LDA, possivelmente irá originar um capítulo no *cookbook* sendo construído.

O LDA caracteriza cada tópico i como uma distribuição de probabilidades sobre os termos. Por exemplo, *threads* relacionadas à Java provavelmente irão conter termos como "object" ou "class", assim esses termos terão alta probabilidade em um possível tópico sobre Java [44]. A Tabela 3.1 mostra para um tópico minerado para a API SWT (biblioteca Java usada para construção de elementos de interface gráfica), a distribuição de probabilidades

[2]http://lucene.apache.org/core
[3]http://htmlcleaner.sourceforge.net

dos 10 top-termos para este tópico. Os números indicam importância relativa (a soma dos valores de todos os termos é 1, mas na tabela como são mostrados apenas os 10 termos mais frequentes para o tópico, a soma é inferior à 1). Assim, o termo "table" é aproximadamente dez vezes mais importante (considerando a frequência) que o termo "checkbox".

Tabela 3.1: 10 top-termos e suas frequências em um tópico do SWT

Termo	Importância Relativa (frequência)
tabl	0,054351167
column	0,025278521
row	0,016221274
cell	0,013537645
tableview	0,011860377
select	0,008617659
viewer	0,007387662
jface	0,006716755
item	0,005710394
checkbox	0,005263123

Adicionalmente, como cada documento pode se referir a vários tópicos (e.g, relacionado a Java, XML e Eclipse), o LDA também define uma distribuição de probabilidade ao longo dos tópicos para cada documento *j*. Essa distribuição define, para cada documento e para cada tópico, a aderência do documento ao tópico. A Tabela 3.2 mostra para um dos documentos presente no corpo de documentos gerados para a API SWT, a aderência deste documento específico em relação a cada um dos tópicos minerados (neste exemplo, foram minerados 15 tópicos, representados por id's de 0 até 14). A soma da aderência do documento em relação a cada um dos tópicos é sempre 1. Observe que neste exemplo o tópico mais aderente ao documento é o tópico de id 0.

Ambas as distribuições (os termos de cada tópico e os tópicos presentes em cada documento), não são sabidas de antemão. O objetivo do LDA é aproximar ambas distribuições através da maximização da probabilidade geral do modelo [44]. As informação destas duas distribuições são usadas no algoritmo de geração de documentos, explicado adiante. Dado um documento, o tópico que possui a maior aderência em relação àquele documento, é chamado de *Tópico Dominante* para o documento. Por exemplo, o *Tópico Dominante* do documento apresentado na Tabela 3.2 é o de id 0.

O número de tópicos (K) é uma parâmetro especificado pelo usuário e provê controle sobre a granularidade dos tópicos descobertos. Valores pequenos geram tópicos mais genéricos e valores maiores geram tópicos mais detalhados. Não existe um único valor de K que é adequado para todas as situações e todos os *datasets* [68]. A análise feita para encontrar um número adequado para este parâmetro é mostrada na Seção 3.5.

Tabela 3.2: Aderência de um documento do corpo de documentos do SWT a cada um dos tópicos minerados pelo LDA

Id do Tópico	Aderência do Documento ao Tópico
0	0,73729902
10	0,13085112
4	0,12831533
8	0,00054619
13	0,00042284
7	0,00040515
2	0,00039764
9	0,00027917
6	0,00025486
3	0,00022783
11	0,00021607
12	0,00017896
5	0,00015485
1	0,00013306

3.4 Algoritmo de Construção de *Crowd Cookbooks*

Suponha que estejamos gerando um *cookbook* para uma determinada API. Pressupõe-se que o LDA já tenha sido aplicado sobre o corpo (*Corpus*) de documentos (*threads*) cujas perguntas são *How-To-Do-It*s, gerando K tópicos. Para cada documento D pertencente ao corpo de documentos, tem-se a aderência que ele possui com cada um dos K tópicos.

Cada tópico encontrado pelo LDA irá, possivelmente, dar origem a um capítulo no *cookbook* sendo construído. Os capítulos serão preenchidos com *Receitas*. Considere que uma receita é um par de pergunta/resposta do corpo de documentos previamente construído, i.e., um par de pergunta/resposta cuja pergunta é um *How-To-Do-It*. Se uma *thread* possui N respostas, existem N receitas para esta *thread*, sendo que todas as receitas são pares de pergunta/resposta com a mesma pergunta, variado apenas na resposta. A razão para considerar uma receita como um par, ao invés da *thread* inteira se deve ao fato de que em uma mesma *thread* podem existir respostas bem e mal avaliadas pela comunidade. Dessa forma, ao considerar os pares separados, é possível incluir no *cookbook* apenas a parcela de maior qualidade da *thread*. O *Score* de um *post* é um *proxy* para a sua qualidade, pois é a principal maneira pela qual a comunidade do SO tem para avaliar o conteúdo do *site*.

Foram elaboradas algumas condições que tornam uma receita (i.e., um par) passível de ser incluída em um *cookbook*:

- **Condição 1**: A resposta deve conter trechos de código-fonte. Os códigos-fonte num *post* são identificados por meio do uso das tags HTML "<pre><code>...". Essa exigência é feita pelo fato de que programar por meio de exemplos é uma maneira intuitiva de aprender tanto para programadores iniciantes quanto experientes [36];

- **Condição 2**: A pergunta e a resposta que compõem o par não devem ter *dead links* em seu conteúdo. Esta é uma maneira de assegurar que as fontes externas referenciadas nos *posts* ainda estão disponíveis para acesso. As URL são identificadas através da presença da *tag HTML* <a href="..." . Para verificar se uma uma URL é um *dead link*, foi utilizada a biblioteca Java HttpUnit[4];
- **Condição 3**: A pergunta não deve ser muito grande. Perguntas com muito conteúdo geralmente contêm muitas subperguntas (i.e., o questionador pergunta muitas coisas), ou têm um problema muito difícil de entender (e.g., um problema muito específico). Decidiu-se não incluir este tipo de pergunta para tornar os *cookbooks* mais fáceis de ler e entender. O *threshold* escolhido para determinar se uma pergunta é grande ou não, é explicado na Seção 3.5.

Para cada *thread* da API cuja pergunta é *How-To-Do-It* são obtidos os pares que satisfazem às três condições descritas acima. Dentre esses pares, o que possuir o maior *score* poderá ser incluído no *cookbook*, dentro do capítulo correspondente ao tópico dominante para o documento ao qual aquele par pertence. Por exemplo, se um par de pergunta/resposta satisfaz às três condições acima, e se ele é o par que que possui o maior *score* (dentre os pares que satisfazem às condições), ele poderá ser incluído no capítulo 7, supondo que tópico de id 7 é o tópico dominante para a *thread* ao qual o par pertence.

Como cada *post* (pergunta ou resposta) individual no SO tem o seu próprio *score* e nós estamos considerando uma receita como um par de pergunta/resposta, é necessário definir uma métrica que indique a qualidade do par como um todo. Uma possível abordagem para isso, é considerar o valor médio do *score* da pergunta e do *score* da resposta de um par. Decidiu-se não seguir essa abordagem porque a resposta parece ser mais importante que a sua respectiva pergunta. A razão para esta hipótese é que a resposta geralmente contém mais informações sobre a solução do problema. Por isso, decidiu-se considerar o *score* de um par como a média ponderada entre os *scores* de sua pergunta e resposta. Arbitrariamente foram definidos os valores 7 e 3 para os pesos dos *scores* da resposta e pergunta, respectivamente. Diferentes valores para estes pesos poderiam levar a diferentes resultados, mas espera-se conseguir resultados aceitáveis com esta escolha.

O **Algoritmo 1** é o pseudo-código para construção de um *cookbook* para uma API A. O objetivo é construir um *cookbook* que possua no mínimo R receitas. O valor definido para o R á apresentado na Seção 3.5 e foi baseado em uma análise de versões comerciais de *cookbooks* existentes. A razão para essa restrição é evitar a construção de *cookbooks* muito pequenos, com tamanho muito inferior aos existentes em versão comercial.

Outra preocupação existente é incluir em um *cookbook* para uma determinada API apenas pares que possuem alto *score* comparativamente ao conjunto de pares da API, pois queremos incluir apenas conteúdo de boa qualidade no *cookbook*. Com esse objetivo, é

[4]http://httpunit.sourceforge.net/

Algoritmo 1: Geração de um *cookbook* C para uma API A

```
maximumPositionAllowed ← getMaximumPositonAllowed(A);
numberOfRecipes ← 0;
while numberOfRecipes < R do
    C ← new Cookbook();
    foreach D ∈ Corpus do
        dominantTopic ← getDominantTopic(D);
        p ← getPair(D);
        if p = null then
            continue;
        adherence ← getAdherence(D, dominantTopic);
        rankingPosition ← getRankingPosition(p);
        if adherence ≥ Ta then
            if rankingPosition ≤ maximumPositionAllowed then
                includePairsInCookbook(dominantTopic, p, C)
                numberOfRecipes ← numberOfRecipes + 1;
    removeSmallChapters(C);
    maximumPositionAllowed ← maximumPositionAllowed + 10;
    numberOfRecipes ← 0;
```

estabelecido um *threshold maximumPositionAllowed* que define a posição máxima que um par deve possuir no ranqueamento dos pares da API (pelo *score* dos pares), para o mesmo estar apto a ser incluído no *cookbook*. Esse ranqueamento considera apenas o melhor par (i.e., o par com maior *score*) de cada *thread*, dentre os pares que satisfazem às condições **Condição 1**, **Condição 2** e **Condição 3**. Considere que a função *getRankingPosition* recupera a posição de um par p neste ranqueamento.

O algoritmo para construção de *cookbooks* é iterativo, pois caso ao final de uma iteração o número de receitas incluídas no *cookbook* seja menor que R, relaxa-se o valor do *threshold maximumPositionAllowed*, para permitir a construção de um *cookbook* com mais receitas, na próxima iteração. Esse relaxamento faz com que a qualidade média das receitas incluídas no *cookbook* em uma iteração seja ligeiramente inferior à qualidade das incluídas na iteração anterior. O valor inicial para o *threshold maximumPositionAllowed* é definido a partir de uma análise do formato de uma curva plotada a partir dos valores do ranqueamento dos pares pelo *score*. A maneira de fazer está análise é explicada na Seção 3.5. Suponha que a função *getMaximumPositionAllowed* retorne o valor inicial deste *threshold* para uma API A.

Para cada documento (*thread*) do corpo de documentos, é recuperado, dentre os pares que respeitam as três restrições, aquele que possui o maior *score*. Suponha que a função *getPair* retorne este par, denotado por p. Pode ser que para um documento, não exista nenhum par que satisfaça as três condições. Nesse caso, a função *getPair* retorna um valor nulo e o algoritmo processa o próximo documento no corpo de documentos. Através

da função *getDominantTopic*, o *Tópico Dominante* da *thread* é recuperado e armazenado na variável *dominantTopic*. A aderência do documento D ao tópico *dominantTopic* é recuperada por meio da função *getAdherence* e armazenada na variável *adherence*. Em seguida é verificado se *adherence* é maior do que um threshold *Ta* definido. Em caso negativo, o par não está apto para ser incluído no *cookbook*. Esse *threshold* foi definido com objetivo de incluir nos capítulos do *cookbook*, apenas receitas fortemente relacionados ao tema do capítulo. O valor definido para esse *threshold* é explicado na Seção 3.5. Em caso positivo, o par é adicionado ao *cookbook*, dentro do capítulo correspondente ao *dominantTopic*. Uma variável chamada *numberOfRecipes* é usada para contar o número de receitas incluídas no *cookbook*.

Ao final do laço utilizado para construir o *cookbook*, é chamado o método *removeSmallChapters* que remove do *cookbook* todos os capítulos com duas ou menos receitas. Isso é feito para evitar capítulos muito pequenos. A ideia de um capítulo de um *cookbook* é agrupar várias receitas relacionadas ao mesmo tempo e por essa razão não faz sentido a existência de capítulos com poucas receitas. Outros valores poderiam ser definidos para determinar se um capítulo é pequeno ou não, mas como esta escolha espera-se construir *cookbooks* de tamanhos aceitáveis (i.e., todos os capítulos terão no mínimo três receitas).

Ao final da construção do *cookbook* verifica-se se *numberOfRecipes* é maior ou igual a *R*. Em caso positivo o algoritmo é encerrado. Em caso negativo, significa que o *cookbook* construído possui um número de receitas inferior ao desejado e por isso, incrementa-se o *threshold maximumPositionAllowed* de 10 e tenta-se construir o *cookbook* novamente, com objetivo de produzir um *cookbook* que possua tamanho superior ao obtido na iteração atual.

O número final de capítulos no *cookbook* pode ser menor do que o número de tópicos (K) usado como *input* para o LDA, porque é possível que para um tópico minerado pelo LDA, nenhum par tenha satisfeito as condições verificadas no algoritmo. Por exemplo, para um tópico minerado pelo LDA, pode ser que tenham sido incluídas apenas duas receitas e por isso este capítulo é removido do *cookbook*.

3.5 Definição de Parâmetros

Nessa seção serão explicados a determinação dos seguintes parâmetros: 1) o número de iteração do LDA; 2) a opção de usar *Bi-Grams* na mineração de tópicos pelo LDA; 3) o número de tópicos (K) minerados pelo LDA; 4) o valor do *threshold* que determina se uma pergunta é grande; 5) o valor do *threshold maximumPositionAllowed*; 6) o valor do *threshold Ta*; 7) o valor do *threshold R*.

Um dos parâmetros do LDA é o número de iterações. Para este parâmetro foi escolhido o valor 2000, pois segundo Griffiths e Steyvers [23], este valor é mais do que o suficiente para o algoritmo de amostragem *Gibbs* se estabilizar.

Um outro parâmetro para o LDA na biblioteca MALLET, é a opção de se usar *Bi-Grams* (i.e., sequências de 2 palavras adjacentes) na mineração dos tópicos [4]. Por exemplo, dado o texto "sort list string", existem três *Uni-Grans* (i.e., termos únicos) ("sort", "list", "string") e dois *Bi-Grams* ("sort_list", "list_string"). Como já foi mostrado que o uso de *Bi-Grams* aumenta a qualidade da análise de texto [62], optou-se por usar tanto *Uni-Grams* como *Bi-Grams* no processo de mineração dos tópicos.

Com o objetivo de identificar um valor adequado para o parâmetro K do LDA, foi feita uma análise com diferentes valores de K, para três diferentes APIs: SWT, LINQ e QT. Essas APIs foram escolhidas por serem populares entre a comunidade de desenvolvimento, e por estarem relacionadas à diferentes linguagens de programação: Java, linguagens .NET e C++, respectivamente.

Foram testados oito valores para o parâmetro K: 5, 10, 15, 20, 25, 30, 35 e 40. Para cada API e para cada valor de K, o LDA foi executado uma vez. Apesar de terem sido testados apenas oito valores para K, foi possível perceber como a aderência dos documentos aos tópicos varia à medida que se muda o valor de K. Como cada tópico possivelmente originará um capítulo no *cookbook* sendo gerado, é desejável que eles tenham documentos (*threads*) com alta aderência, pois serão esses documentos que originarão as receitas a serem incluídas nos capítulos.

Após cada execução do LDA, para i variando de 1 até 200, calculou-se a média, dentre todos os tópicos, da aderência dos documentos que possuem a *i-ésima* maior aderência à cada tópico. Por exemplo, se $i = 2$ foi calculada a aderência média de todos os documentos que são os segundos mais aderentes a cada tópico. Foram plotados os gráficos mostrados nas Figuras 3.5, 3.6 e 3.7, para as as APIs SWT, LINQ e QT respectivamente. Os três gráficos possuem comportamento semelhante. Considerando um K fixo (e.g., $K = 5$), obviamente, a medida que se aumenta o valor de i, a aderência média diminui, pois o valor em cada posição i representa a média dos *i-ésimos* documentos mais aderentes a cada um dos tópicos (e.g., a média dos documentos mais aderentes ($i = 1$) é maior do que média dos segundos documentos mais aderentes ($i = 2$)). Para $K = 5$, cada posição do gráfico representa a média de 5 valores (uma vez que existem 5 tópicos neste caso). Para cada gráfico, ao se comparar as curvas associadas a cada valor de K, percebe-se que a medida que aumenta-se o valor de K, a aderência média diminui, ou seja, com pequenos valores de K, o LDA minera tópicos com documentos altamente aderentes e para grandes valores de K, os tópicos gerados possuem documentos com aderência inferior. Outro comportamento que se percebe é que a diminuição na aderência entre as curvas, diminui para maiores valores de K (e.g, a queda de aderência observada ao se variar o K de 5 para 10 é superior à queda observador quando se muda de 35 para 40). A impressão que se tem é que tende a ocorrer uma estabilização nos valores observados, à medida que se aumenta o K, pois as curvas vão se aproximando. Observe que o gráfico para o SWT possiu curvas com formato ligeiramente diferente do formato das curvas dos outros

dois gráficos, especialmente para $K = 5$. Uma possível explicação para este fato é que o número de documentos dos corpo do SWT é bem menor do que no corpo das duas outras APIs (864 documentos, contra 10513 do LINQ e 6904 para o QT).

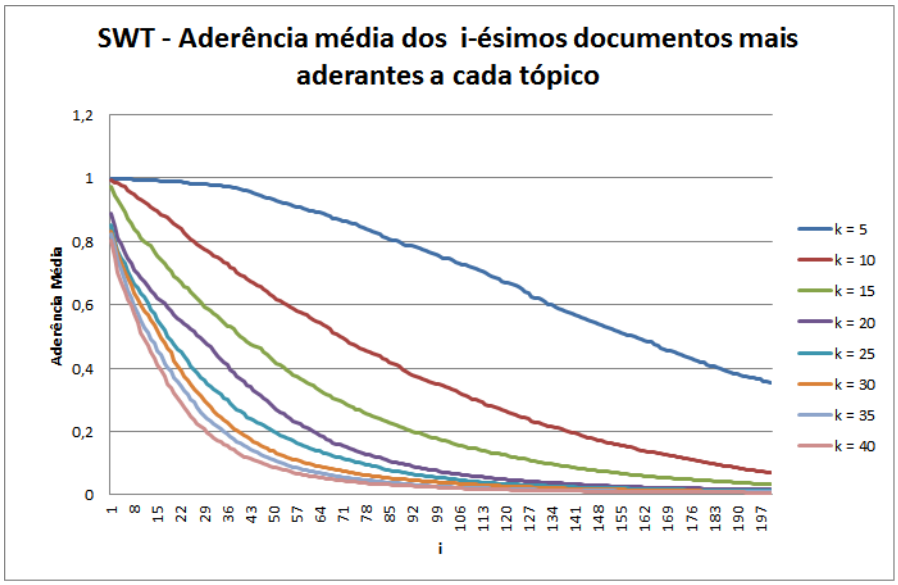


Figura 3.5: SWT - Aderência média dos i-ésimos documentos mais aderantes a cada tópico

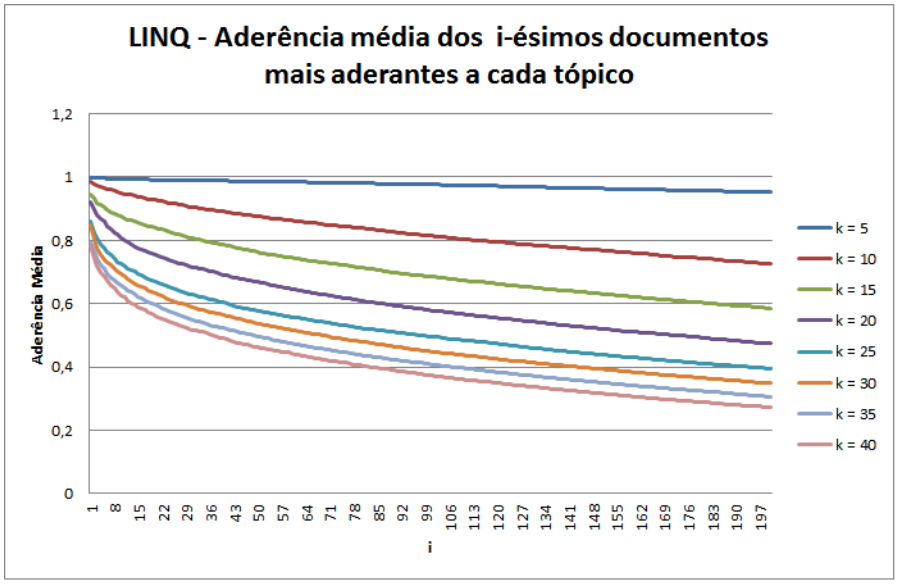


Figura 3.6: LINQ - Aderência média dos i-ésimos documentos mais aderantes a cada tópico

Com o objetivo de produzir *cookbooks* com um número de capítulos parecidos com os existentes em verão comercial, foi selecionada uma amostra de dez *cookbooks*, relacionados a vários assuntos de programação e contou-se o número de capítulos de cada um. Essa informação é mostrada na Tabela 3.3. O número médio de capítulos na amostra analisada é 14,3. Com o objetivo de minerar tópicos com documentos com alta aderência aos tópicos e ao mesmo tempo produzir *cookbooks* com número de capítulos semelhante aos existentes

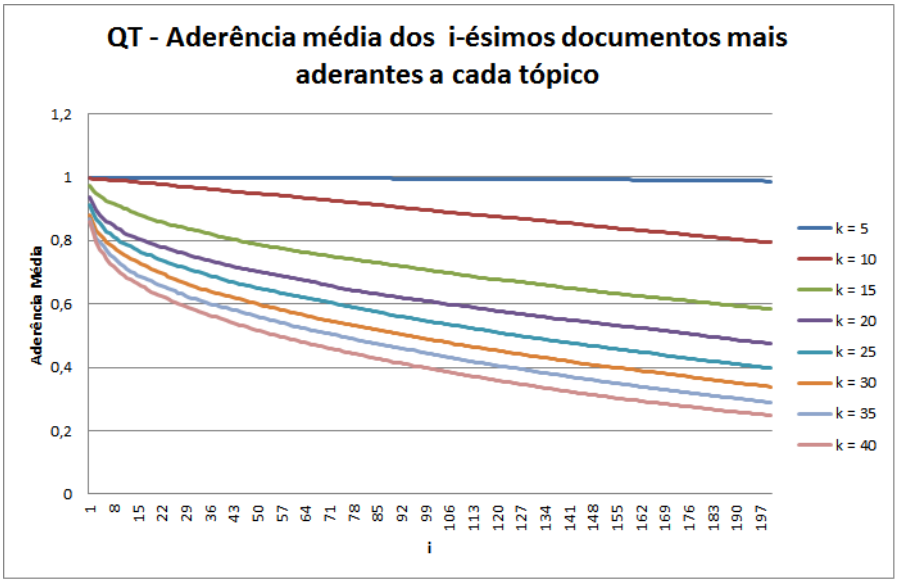


Figura 3.7: QT - Aderência média dos i-ésimos documentos mais aderantes a cada tópico

comercialmente, optou-se por escolher o valor de K igual a 15. Apesar de que o valor 5 foi o que gerou tópicos com documentos mais aderentes, ele não foi escolhido por ser muito inferior ao tamanho observado dos *cookbooks* comerciais. Em contrapartida, apesar de o valor 15 não gerar tópicos com documentos tão aderentes (ainda sim com a terceira maior dentre os valores analisados), ele se assemelha mais ao tamanho dos *cookbooks* comerciais e por isso foi escolhido.

Tabela 3.3: Número de capítulos e receitas em versões comerciais de *cookbooks*

Nome do *Cookbook*	Número de Capítulos	Número de Receitas
Python *Cookbook* [1]	17	245
Jquery *Cookbook* [38]	18	182
Java *Cookbook* [18]	26	299
Boost C++ Application Development *Cookbook* [50]	12	95
C++ *Cookbook* [57]	15	171
The Core iOS 6 Developer's *Cookbook* [54]	13	106
R Graphics *Cookbook* [12]	15	156
OpenGL 4.0 Shading Language *Cookbook* [69]	9	64
Selenium Testing Tools *Cookbook* [66]	11	91
Apache Solr 4 *Cookbook* [34]	7	106

Para a definição do valor para o *threshold* que determina se uma pergunta é grande ou não, foi escolhido um conjunto de 10 APIs representativas de várias áreas do desenvolvimento do software: AWT, Backbone.js, Boost, JQuery, LINQ, Log4Net, QT, STL, Swing e SWT. Para essas APIs foram identificadas as *threads* do tipo *How-To-Do-It* e contou-se o número de caracteres presentes no corpo da pergunta destas *threads*. Em seguida foi plotado o gráfico mostrada na Figura 3.8. Esse gráfico mostra para 14 intervalos de tamanhos de caracteres, o número de perguntas em cada um destes intervalos. Observe que a maior pergunta da amostra analisada possui 26.900 caracteres. 82,53% (98.664) das perguntas possui tamanho no máximo de 1.299 caracteres. Por essa razão, decidiu-se escolher o valor deste *threshold* igual a 1300 caracteres. Com esta escolha, a maior parte das perguntas é mantida (na amostra analisada, apenas 17,47% das perguntas seria

descartada), o que não deve prejudicar significativamente os *cookbooks* produzidos, pois espera-se que existam pares de boa qualidade dentre o percentual que não é descartado, visto que mais de 80% dos *threads* seriam mantidas. Assim, caso a pergunta de um *thread* possua tamanho maior ou igual a 1300 caracteres, os pares pertencentes à esta *thread* não poderão fazer parte de *cookbooks*.

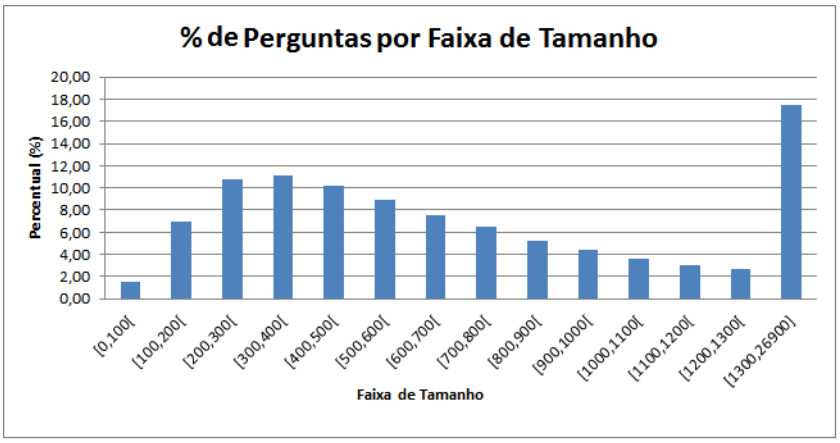


Figura 3.8: Percentual de perguntas por faixa de tamanho

Para definir o valor inicial do *threshold maximumPositionAllowed* deve-se fazer uma análise separada por API. Para cada API, seleciona-se as *threads* cuja pergunta é *How-To-Do-It* e faz-se um ranqueamento dos pares da API pelo score dos mesmos. Esse ranqueamento considera apenas o melhor par (i.e., o par com maior *score*) de cada *thread* que satisfaz as três condições estabelecidas na Seção 3.4. Para as APIs SWT, LINQ e QT foram construídos gráficos (Figuras 3.9 e 3.10, 3.11) que mostram para cada par (cada posição no eixo-x) o seu *score* (posição no eixo-y). A ideia nesta análise é observar o formato da curva e escolher um ponto de corte no eixo-x para o qual os pares à esquerda deste ponto possam ser considerados a "elite" da API, i.e., os melhores pares da API (os pares que claramente possuem *score* alto em relação à totalidade de pares da API). Observe que esta análise deve ser feita para cada API, uma vez que o número de *threads* pode variar bastante entre APIs, o que faz com que o ponto de corte escolhido seja potencialmente diferente para cada API. No caso das APIs SWT, LINQ e QT, os pontos de corte escolhidos foram 40, 200 e 200 respectivamente e foram destacados com uma linha vermelha vertical nos gráficos das Figuras 3.9, 3.10 e 3.11. Observe que os pares à esquerda da linha vermelha podem ser considerados a elite de suas API e que à direita deles tende a ocorrer uma estabilização nos *scores* dos pares. O fato da escolha deste pontos de corte ter de ser feita manualmente, torna a abordagem para construção de *cookbooks* semi-automatizada.

Para definição do valor do *threshold Ta*, que define a aderência mínima que uma *thread* deve possuir com seu tópico dominante para os pares da *thread* estarem aptos a serem incluídos como receitas nos *cookbooks*, foram construídos *cookbooks* para as APIs SWT, LINQ e QT sem considerar esta restrição (nesse caso particular, está sendo usando K =

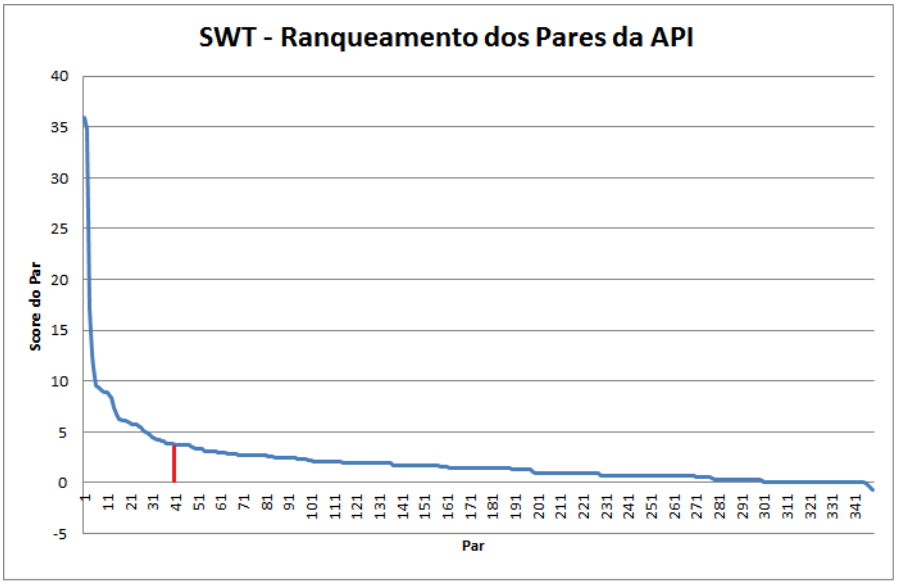


Figura 3.9: Ranqueamento dos pares da API SWT

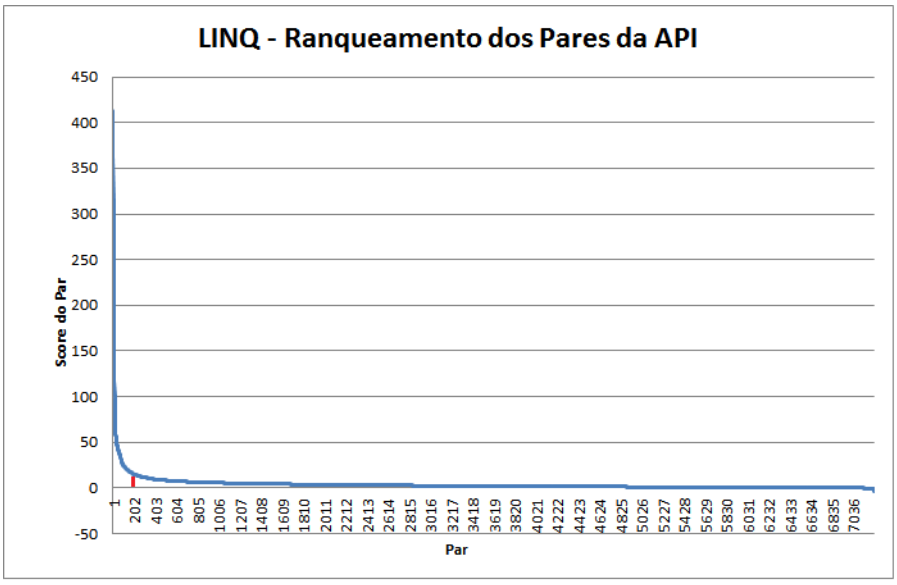


Figura 3.10: Ranqueamento dos pares da API LINQ

10). Considere que a aderência de uma receita *r1* (i.e., um par de pergunta/resposta) à um tópico *tp1* é igual a aderência da *thread* que originou *r1* à *tp1*. Foram plotados os gráficos das Figuras 3.12, 3.13 e 3.14, que mostram, para cada tópico (i.e., cada curva dos gráficos) a aderência que as *threads* dos pares incluídos nos capítulo possuem com relação ao tópico do LDA que deu origem ao capítulo no qual a receita foi incluída. Considerando uma curva (i.e., um tópico), a aderência dos pares foi ordenada de maneira decrescente. É possível perceber que existem receitas com aderência baixa a seus tópicos, i.e., receitas cujos termos não coincidem fortemente como os principais termos minerados para o tópico. Por exemplo, para SWT, LINQ e QT, existem, respectivamente, 6, 25 e 7 receitas com aderência inferior a 0.4. Não é interessante incluir nos *cookbooks* esse tipo de receitas, pois

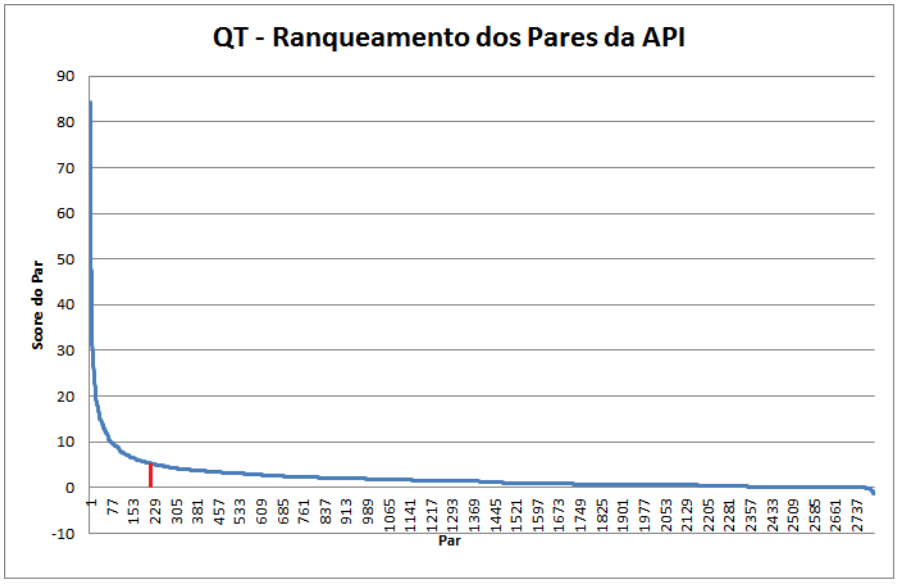


Figura 3.11: Ranqueamento dos pares da API QT

em um *cookbook*, as receitas de um capítulo devem estar fortemente relacionados com o termos presentes no título do capítulo (e.g., em um capítulo sobre "Strings" devem estar presentes apenas receitas que tratam do assunto "String"). Decidiu-se por escolher o valor do *threshold* Ta igual 0.51, o que significa que pelo 51% dos termos presentes em uma receita *r1* devem estar relacionados a um tópico *tp1* para a mesma poder ser incluída no capítulo do *cookbook* correspondente à *tp1*. Dessa forma, garante-se que a maior parte dos termos das receitas estão relacionados aos capítulos.

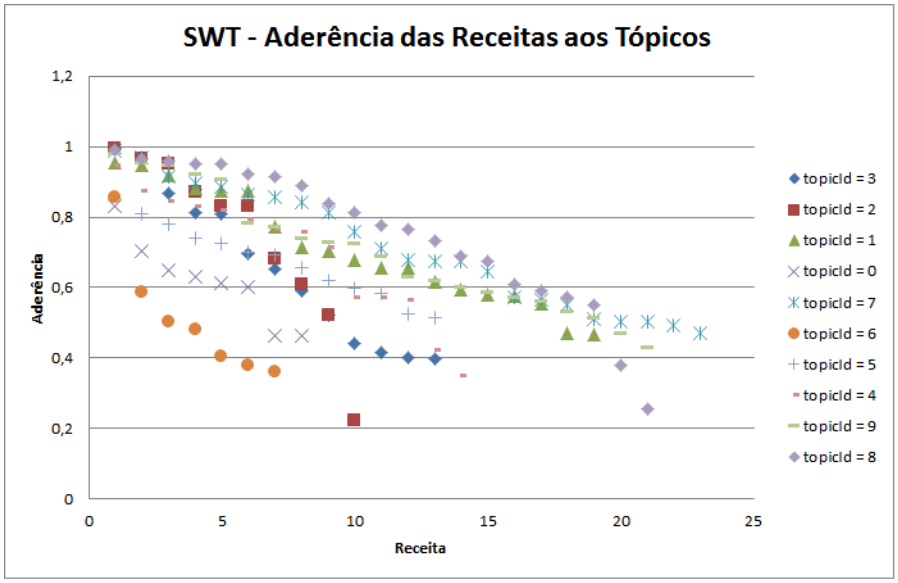


Figura 3.12: Aderência das receitas as tópicos para o SWT

Para definir o valor do *threshold* R que determina o número mínimo de receitas que um *cookbook* deve conter, utilizou-se a informação do número de receitas presentes na amostra de dez *cookbooks* mostrada na Tabela 3.3. Como o menor *cookbook* desta tabela possui

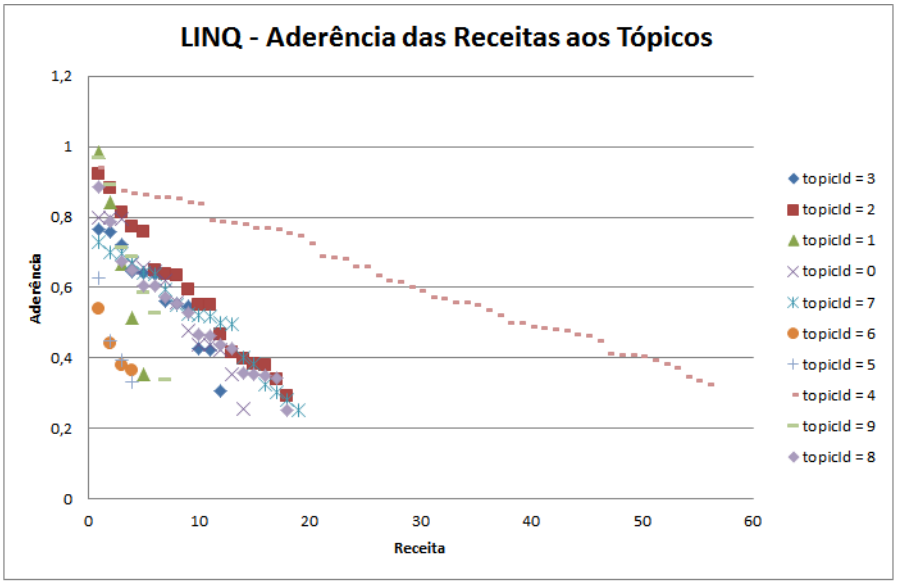


Figura 3.13: Aderência das receitas as tópicos para o LINQ

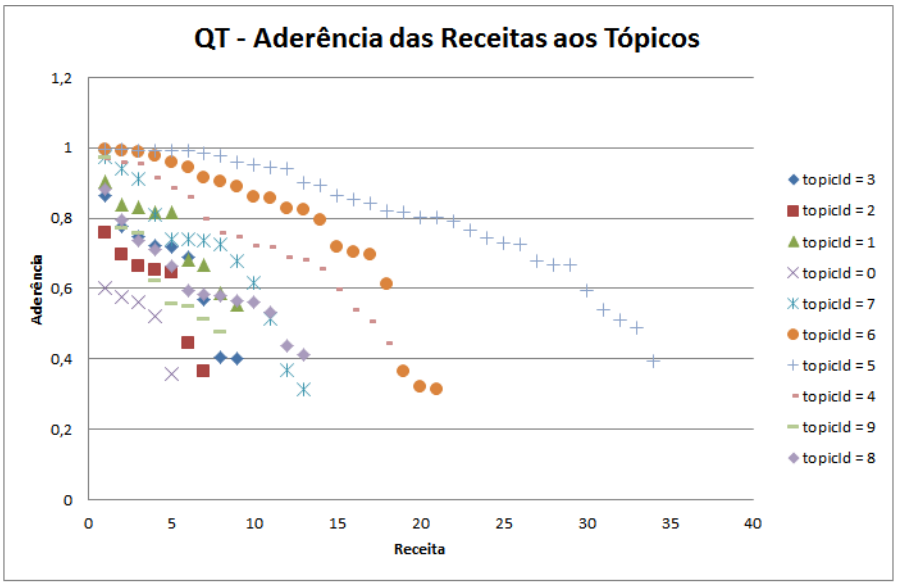


Figura 3.14: Aderência das receitas as tópicos para o QT

64 receitas, optou-se por escolher este valor para o *threshold* R. Assim, a abordagem de construção de *cookbooks* irá gerar *cookbooks* de no mínimo 64 receitas.

3.6 Aplicação *Web* para visualização dos *Crowd Cookbooks*

Com o objetivo de tornar os *crowd cookbooks* produzidos pela estratégia proposta disponíveis para a visualização pelo público, foi desenvolvida uma aplicação *web*[5]. Os *cookbooks* são armazenados em um banco de dados relacional e lidos pela aplicação durante a inicialização da mesma. A tela inicial da aplicação é apresentada na Figura 3.15.

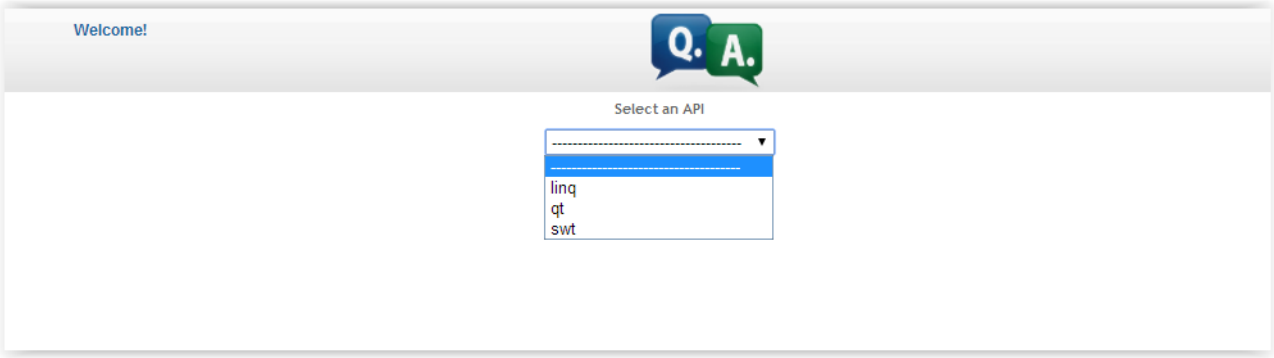


Figura 3.15: Tela inicial da aplicação Web

O usuário tem a opção de selecionar qual API ele deseja visualizar o *cookbook*. Uma vez que ele escolheu uma API, abre-se um nova janela (Figura 3.16), na qual é mostrada a estrutura de capítulos do *cookbook* para a API escolhida. Cada capítulo é identificado por um id (e.g., o id do primeiro capítulo é "1", por isso é mostrado "Chapter 1") que corresponde ao id gerado para o tópico no LDA (como o parâmetro K para o LDA possui valor igual a 15, o LDA minera tópicos com id's variando de 0 até 14). Cada capítulo também é identificado pelos 5 termos mais representativos do tópico minerado pelo LDA (e.g., o "Chapter 2" é representado pelos termos "tabl column row cell tableview"). Ao clicar sobre o título de algum dos capítulos, a tela se expande (Figura 3.17) para mostrar os títulos das receitas (i.e., o título da pergunta que compõe o par de pergunta/resposta) que estão contidas dentro do capítulo. Cada receita também é identificada por um id composto de dois números separados por um ".". O primeiro número coincide com o id do capítulo e o segundo simplesmente é a posição da receita dentro do capítulo (e.g., a primeira receita dentro do "Chapter 2", possui id "2.0").

Ao clicar sobre o título de alguma receita, abre-se uma nova tela que mostra as informações relativas à receita (Figura 3.18). Nesta tela, o usuário pode visualizar (a) o título do capítulo onde a receita se encontra; (b) o título da receita, (c) as *tags* pertencentes à pergunta; (d) o corpo da pergunta; (e) o corpo da resposta. Caso queria, o usuário pode clicar no botão "Back" (f) para voltar à tela que mostra a estrutura de capítulos do *cookbook*. Através do botão "«Previous Recipe" (g), o usuário pode navegar para a

[5]http://lascam.facom.ufu.br:8080/QAWeb_dis/

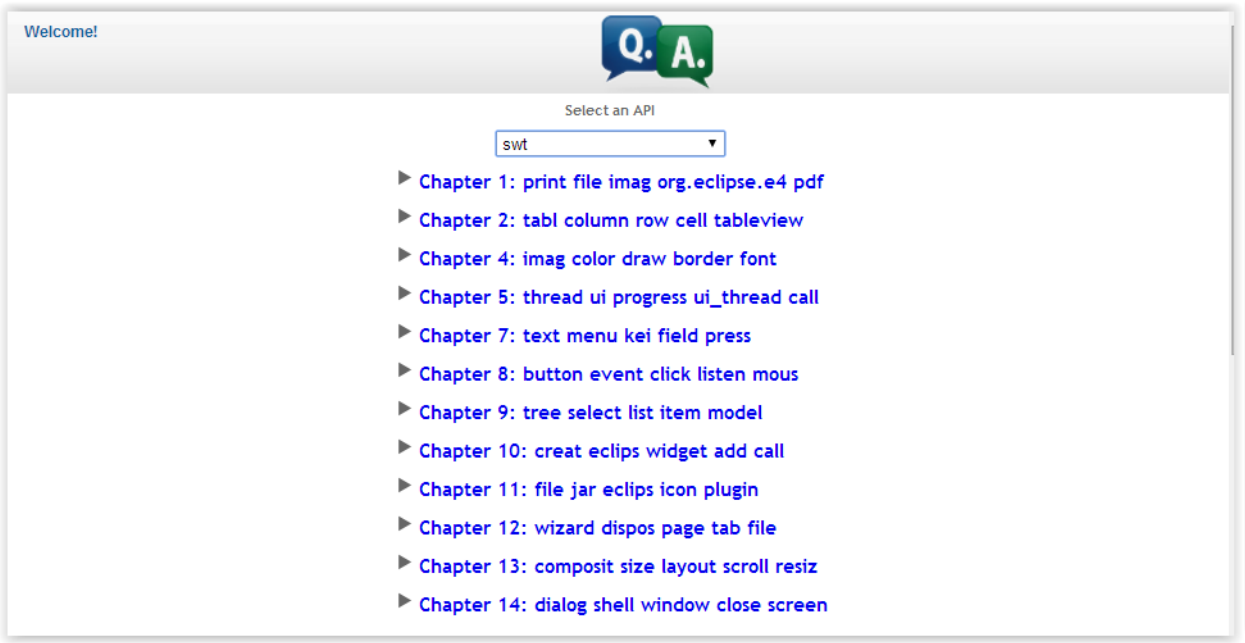


Figura 3.16: Tela que mostra a organização em capítulos do *cookbook* sobre SWT

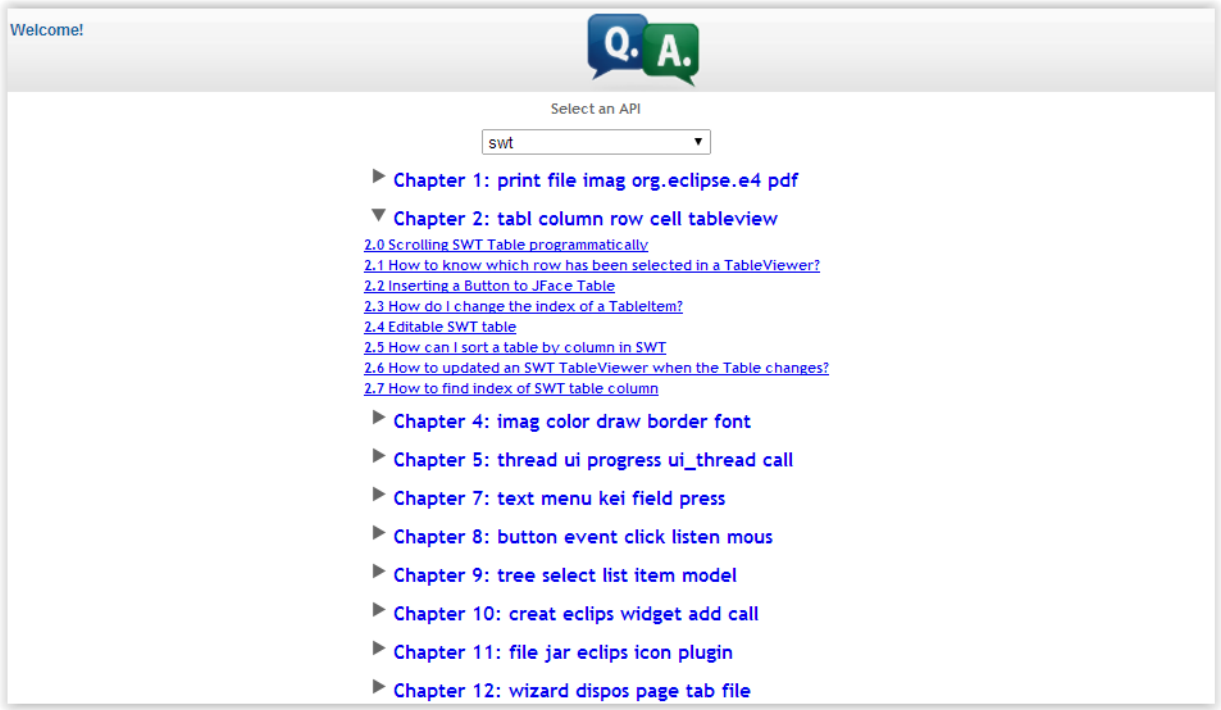


Figura 3.17: Tela que mostra um capítulo expandido no *cookbook* sobre SWT

receita anterior àquela que está sendo visualizada no momento, e ao clicar no botão "Next Recipe»" (h), o usuário passa para a próxima receita no *cookbook*.

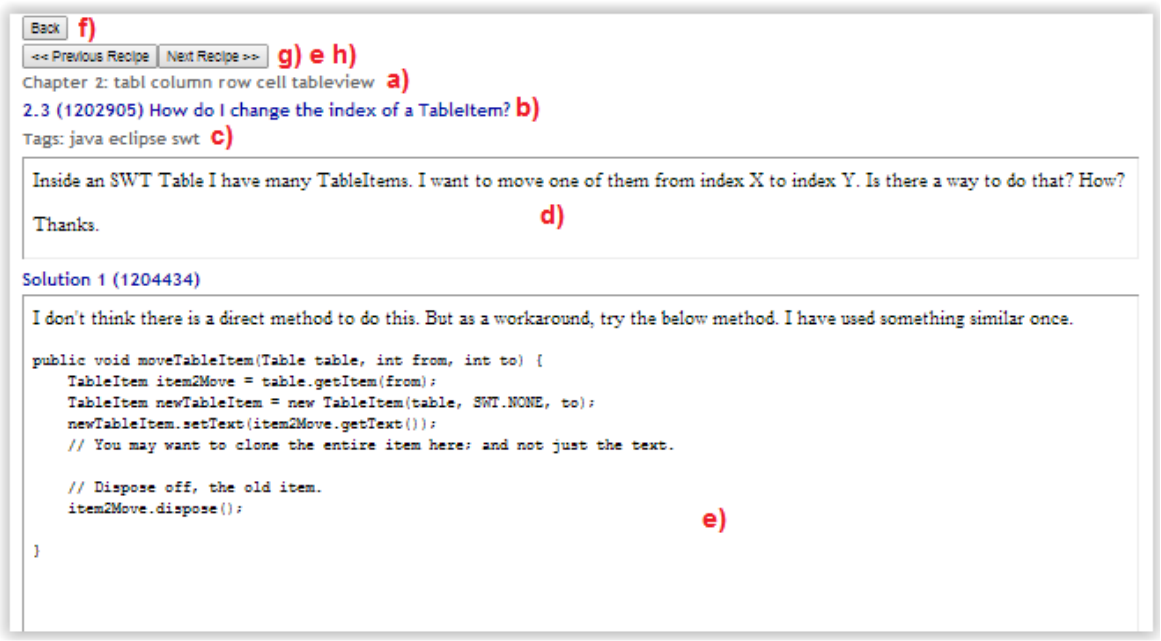


Figura 3.18: Tela que mostra uma receita

Sumarização da Abordagem para Construção de *Crowd Cookbooks*

Neste capítulo foi apresentada uma abordagem para construção de *cookbooks* a partir do *crowd knowledge* do SO. Os passos da estratégia apresentada são: 1) Deve-se selecionar todas as *threads* de uma determinada API no banco de dados do SO; 2) Dentre as *threads* selecionadas no passo anterior, deve-se identificar quais delas possuem pergunta que é um *How-To-Do-It*; 3) Em seguida, deve-se criar um corpo de documentos composto pelas *threads* selecionadas no passo anterior, e pré-processar os documentos deste corpo; 4) Aplica-se o LDA para identificar os potenciais capítulos do *cookbook* sendo construído; 5) Em seguida, deve-se identificar as receitas que serão incluídas no *cookbook* sendo construído e consequentemente, os capítulos do *cookbook*. Neste capítulo também foi descrito um estudo para a definição do valor de vários *threshols* e parâmetros usados ao longo da abordagem. Uma aplicação *web* foi construída para permitir que os *cookbooks* construídos pela abordagem proposta possam ser visualizados pelo público.

Capítulo 4

Metodologia para Avaliação de *Crowd Cookbooks*

Neste capítulo será apresentada a metodologia proposta para avaliação dos *Crowd Cookbooks*. As perguntas de pesquisa do estudo são enunciadas na Seção 4.1. Na Seção 4.2 são apresentados alguns critérios que foram definidos para algumas das perguntas de pesquisa definidas. A Seção 4.3 descreve as APIs consideradas no estudo e os *crowd cookbooks* construídos para as mesmas. Na Seção 4.4 são apresentadas informações a respeito dos sujeitos humanos recrutados para participar do estudo. A Seção 4.5 descreve os procedimentos utilizados para desenvolver o estudo, como por exemplo, os questionários desenvolvidos para avaliação dos *cookbooks* e o processo de amostragem para seleção de perguntas e receitas para serem avaliadas pelos participantes.

4.1 Perguntas de Pesquisa

Foi desenvolvido um estudo para avaliar a abordagem apresentada nesta dissertação. Inicialmente, o estudo irá verificar se os *crowd cookbooks* produzidos pelas abordagem proposta contêm algumas propriedades desejáveis de *cookbooks*. Foram definidos seis propriedades desejáveis de *cookbooks*, e para cada um dessas propriedades existe uma pergunta de pesquisa associada. As perguntas de pesquisa e propriedades desejáveis são descritas na sequência.

A primeira propriedade diz respeito à **semântica dos capítulos**. Cada capítulo de um *cookbook* deve possuir um tema bem definido e deve ser possível identificar esse tema a partir da leitura do título do capítulo. Em relação à esta propriedade, foi elaborada a pergunta de pesquisa **RQ1: Em que extensão os capítulos dos *crowd cookbooks* possuem sentido definido?**

Um outra propriedade desejável dos *cookbooks* é que não exista a **sobreposição de temas** entre capítulos. Assim, por exemplo, em um *cookbook* sobre Java não deve existir

dois capítulos sobre "Manipulação de Strings". Em relação à esta propriedade, foi elaborada a pergunta de pesquisa **RQ2: Em que extensão existe a sobreposição de temas entre os capítulos dos *crowd cookbooks*?**

Outra propriedade desejável dos *cookbooks* é que as receitas dentro de um capítulo estejam **relacionadas ao tema do capítulo**. Obviamente, se uma receita está dentro de um capítulo sobre "Manipulação de Strings", ela deve tratar deste tema. Para esta propriedade, foi definida a pergunta de pesquisa **RQ3: Em que extensão as receitas dos *crowd cookbooks* estão relacionadas com o tema do capítulo onde se encontram?**

Outra propriedade desejável dos cookooks é que os pares de pergunta/resposta que são incluídos como receitas neles, de fato são **adequados** para serem parte daquele *cookbook*. Receitas são pares de perguntas/resposta em que a pergunta é um *How-To-Do-It*. Portanto, pares em que a pergunta é de outro tipo, como conceitual ou em que o questionador pede ajuda para correção de um *bug* não são adequadas para estarem em um *cookbook*. Para esta propriedade, foi definida a pergunta de pesquisa **RQ4: Em que extensão as receitas dos *crowd cookbooks* são adequadas para fazerem parte de *cookbooks*?**

A próxima característica desejada é a **auto-contenção** das informações contidas nas receitas. Muitas receitas possuem *links* para fontes externas (exemplo: *site* da documentação oficial da API, *blogs*, fóruns). Apesar desse tipo de conteúdo ser benéfico no sentido servir como um complemento à solução apresentada no *cookbook*, não é desejado que a solução esteja presente apenas nesta fonte externa, pois não existe nenhuma garantia que as soluções apresentadas nestas fontes fiquem disponíveis para sempre (um dia esses sites podem "sair do ar"). Assim, é importante que as informações contidas em um *cookbook* estejam auto-contidas, ou seja, mesmo que existam referências a *sites* externos, as informações presentes no texto da receita devem ser suficientes para o entendimento do problema e solução lá apresentados. Para esta propriedade, foi definida a pergunta de pesquisa **RQ5: Em que extensão as receitas dos *crowd cookbooks* possuem informações auto-contidas?**

A última propriedade desejada para os *cookbooks* é que os trechos de código-fonte presentes nas receitas sejam **reproduzíveis**, i.e., que eles possam diretamente compilados e executados usando um compilador ou um ambiente de programação como um IDE (exemplo: Eclipse), depois de um simples "copiar e colar". Para esta propriedade, foi definida a pergunta de pesquisa **RQ6: Em que extensão os trechos de código-fonte das receitas dos *crowd cookbooks* são reproduzíveis?**.

Outro objetivo do estudo é verificar se as opiniões dos sujeitos humanos em relação às propriedades desejáveis dos coobooks se assemelham. Assim, poderemos verificar em que extensão existe concordância entre os avaliadores do estudo em relação às propriedades desejáveis. Para este objetivo, foi definida a pergunta de pesquisa **RQ7: Em que extensão as opiniões dos sujeitos humanos em relação às propriedades "semântica**

do capítulo", "relacionamento com tema do capítulo", "adequabilidade para ser parte de *cookbook*", "auto-contenção" e "reprodutibilidade dos códigos-fonte" se assemelham?

Outro objetivo de estudo é verificar a quantidade de receitas que atendem à mais de um propriedade desejável ao mesmo tempo. O ideal é que as receitas atendam à mais de uma propriedade, pois cada uma corresponde à uma vantagem diferente. Das propriedades desejáveis de *cookbooks*, quatro são relacionadas a receitas (**relacionamento com tema do capítulo, adequabilidade para ser parte de *cookbook*, auto-contenção e reprodutibilidade dos códigos-fonte**). Assim, em relação a este objetivo de pesquisa foi definida a pergunta de pesquisa **RQ8: Qual a quantidade de receitas que atendem simultaneamente às propriedades "relacionamento com tema do capítulo", "adequabilidade para ser parte de *cookbook*", "auto-contenção" e "reprodutibilidade dos códigos-fonte"?**

O próximo objetivo do estudo é identificar a percepção dos participantes do experimento em relação aos pontos positivos, pontos negativos e oportunidades de melhoria dos *crowd cookbooks*. Para este objetivo, foi definida a pergunta de pesquisa **RQ9: Quais são os pontos positivos, pontos negativos e melhorias que podem ser feitas nos *crowd cookbooks*?**

O último objetivo do estudo é identificar a percepção dos participantes em relação à utilidade dos *crowd cookbooks* em relação ao aprendizado de APIs. Para esse objetivo, foi definida a pergunta de pesquisa **RQ10: Qual o papel que os *crowd cookbooks* podem desempenhar no aprendizado de APIs?**

4.2 Critérios para Avaliação dos *Cookbooks*

Para mensurar as propriedades desejáveis de *cookbooks*, explicadas na seção anterior, foram definidos alguns critérios. A avaliação conduzida com os sujeitos humanos servirá para verificar se os *cookbooks* construídos através da abordagem proposta possuem essas propriedades e se dará por meio da atribuição de notas aos critérios definidos. Na construção dos *cookbooks*, optou-se por representar o título de cada capítulo pelos cinco termos mais importantes encontrados pelo LDA para o tópico que deu origem ao capítulo.

Para verificar se os *cookbooks* produzidos pela estratégia apresentada possuem as propriedades que foram definidas, foi conduzido um estudo com sujeitos humanos no qual eles foram solicitados a avaliar parte dos *cookbooks* gerados, conforme os critérios que serão descritos nesta seção. Todos critérios possuem uma escala *Likert* com cinco valores, variando de 2 (melhor valor) até -2 (pior valor). Para facilitar a avaliação pelos sujeitos humanos e criar uma certa padronização entre as avaliações das várias pessoas, para cada valor foi escrita uma frase explicativa para servir como um guia durante as avaliações. O valor 0 foi reservado para os casos em que o avaliador não soube avaliar um critério.

Para que seja possível quantificar a propriedade **semântica dos capítulos**, foi definido o critério **Semântica do Capítulo (*Smt*)**. Este critério tem o objetivo de avaliar se é possível derivar um significado para o título do capítulo a partir de seus cinco termos representativos. A nota para esse critério varia de 2 a -2. Os valores desta escala foram definidos e expostos aos sujeitos humanos da seguinte forma:

- (2) É possível atribuir um único significado (tema) preciso ao capítulo. Todos os cinco termos do título estão relacionados a este tema;
- (1) É possível atribuir um significado ao capítulo, porém algum(s) dos cinco termos não está(ão) relacionados ao tema encontrado;
- (0) Não soube opinar ou identificou mais de um tema;
- (-1) É possível perceber algum relacionamento entre os termos contidos no título, mas você não consegue atribuir um significado (tema) para este título;
- (-2) Não é possível perceber nenhum tipo de relacionamento entre os termos contidos no título.

A Seção 5.3 mostra os resultados e análise referentes à avaliação do critério *Smt*.

Para mensurar a propriedade **relacionadas ao tema do capítulo**, foi definido o critério **Relacionamento com o Capítulo (*Relac*)**. O objetivo deste critério é avaliar o relacionamento de cada receita com as 5 palavras-chave de seu capítulo, uma vez que é a partir da leitura destas palavras-chaves que é possível perceber o possível tema do capítulo. Os valores para este critério variam de 2 a -2 e são explicados a seguir:

- (2) Totalmente relacionada: A receita está relacionada com três ou mais termos presentes no título do capítulo;
- (1) Parcialmente relacionada: A receita está relacionada com dois termos presentes no título do capítulo;
- (0) Não soube opinar;
- (-1) Parcialmente não-relacionada: A receita está relacionada com apenas um dos termos presentes no título do capítulo;
- (-2) Totalmente não-relacionada: A receita não está relacionada com nenhum dos termos presentes no título do capítulo.

A Seção 5.5 mostra os resultados e análise referentes à avaliação do critério *Relac*.

Para mensurar a propriedade de **adequabilidade**, foi definido o critério **Adequação para ser parte de um *cookbook* (*Adeq*)**. Nesse caso preferimos dar mais liberdade à interpretação de cada valor porque a noção de adequabilidade é algo bastante subjetivo.

- (2) Totalmente adequada;

- (1) Parcialmente adequada;
- (0) Não soube opinar;
- (-1) Parcialmente inadequada;
- (-2) Totalmente inadequada.

Os resultados e análise referentes à avaliação do critério *Adeq* são abordados na Seção 5.6.

Para mensurar a propriedade de **auto-contenção**, foi definido o critério **Auto-contenção (*AutoCont*)**. Os valores para esse critério variam de 2 a -2:

- (2) Totalmente auto-contida: É possível entender completamente o cenário e solução apresentados na receita apenas considerando-se o conteúdo textual da receita (sem acessar os possíveis *sites* referenciados por ela via *links*, caso eles existam);
- (1) Parcialmente auto-contida: É possível entender a maior parte do cenário e solução apresentados na receita apenas considerando-se o conteúdo textual da receita (sem acessar os possíveis *sites* referenciados por ela via *links*, caso eles existam);
- (0) Não soube opinar;
- (-1) Parcialmente não auto-contida: Para entender a maior parte do cenário e solução apresentados na receita é preciso acessar o conteúdo externo por ela referenciado (exemplos: *sites* externos);
- (-2) Totalmente não auto-contida: As informações contidas na receita são completamente dependentes de fontes externas referenciadas na receita (exemplo: *links* para outros *sites*).

Os resultados e análise referentes à avaliação do critério *AutoCont* são abordados na Seção 5.7.

Para mensurar esta propriedade de **reprodutibilidade**, foi definido o critério **Reprodutibilidade dos códigos-fontes (*Reprod*)**, também variando de -2 a 2:

- (2) O(s) trecho(s) pode(m) ser compilado(s) e executado(s) sem nenhuma alteração;
- (1) O(s) trecho(s) pode(m) ser compilado(s) e executado(s) depois de pequena(s) alteração(ões);
- (0) Não soube opinar;
- (-1) O(s) trecho(s) pode(m) ser compilado(s) e executado(s), porém para isso seria(m) necessária(s) alteração(ões) de médio a grande porte;
- (-2) Não é possível executar o(s) trecho(s) de código-fonte presente na receita ou ela não possui código-fonte.

A partir dos valores definidos para cada critério, percebe-se que os valores 2 e 1 são avaliações positivas (boas), o valor 0 corresponde à uma avaliação neutra e o valores -1 e -2 correspondem à avaliações negativas (ruins). Mesmo com a definição dos significados dos valores, o processo de avaliação é subjetivo. Por exemplo, no caso do critério **Semântica do Capítulo**, um avaliador pode encontrar dois temas para um capítulo, enquanto que outro pode não encontrar nenhum. Ou seja, a subjetividade é algo inerente à maior parte das avaliações feitas por humanos. Porém, com a definição dos valores da escala *Likert*, apesar de ainda existir espaço para a subjetividade, espera-se criar um direcionamento para as pessoas durante o processo de avaliação.

4.3 APIs Consideradas e *Crowd Cookbooks* Construídos

As APIs usadas no estudo apresentado nesta dissertação, para as quais foram gerados *crowd cookbooks*, são:

- SWT[1] é um toolkit para *widgets* da linguagem Java desenvolvido para prover acesso portável e eficiente aos elementos de interface gráfica dos sistemas operacionais;
- QT[2] é uma API multiplataforma para desenvolvimento de interfaces gráficas em C++. Com ele é possível desenvolver aplicativos e bibliotecas uma única vez e compilá-los para diversas plataformas sem que seja necessário alterar o código fonte;
- LINQ[3] (Language Integrated Query): é uma API para o *framework* Microsoft .NET, que adiciona capacidades de consulta às linguagens .NET. Essa API provê extensões curtas e de sintaxes expressivas para manipular dados. O LINQ pode ser considerado uma extensão das linguagens .NET, pois adiciona às mesmas um conjunto de operadores para a manipulação de dados. Dessa forma, o LINQ não é uma biblioteca clássica, como o SWT, pois provê um conjunto de operadores, ao contrário das bibliotecas tradicionais que disponibilizam um conjunto de classes e métodos. A interface pública do LINQ (i.e., sua API) é constituída basicamente por um conjunto de operadores que permitem a manipulação de dados provenientes de diversas fontes (e.g., bancos de dados, *arrays*, XMLs).

Essas APIs foram escolhidas pelo fato de estarem ligadas a diferentes linguagens de programação (Java, linguagens .NET e QT), o que de certa forma nos permite testar a generalidade da abordagem proposta. Além disso, elas são APIs de expressão na área de desenvolvimento de software.

[1] http://www.eclipse.org/swt/
[2] http://qt-project.org/
[3] http://msdn.microsoft.com/pt-br/library/bb397926.aspx

A Tabela 4.1, mostra para as três APIs escolhidas, o número de perguntas (que é igual ao número de *threads*, pois cada *thread* tem uma pergunta), o número de perguntas que foram classificadas como *How-To-Do-It*, e o número de pares de pergunta/resposta que foram geradas a partir das *threads* com pergunta *How-To-Do-It*.

Tabela 4.1: Métricas sobre as APIS selecionadas

API	# de Perguntas	# de Perguntas *How-To-Do-It*	# de Pares *How-To-Do-It*
SWT	2698	864	1228
LINQ	32271	10513	23437
QT	23983	6904	10585

A Tabela 4.2, mostra para os *cookbooks* gerados para as três APIs escolhidas, o número de capítulos e o número de receitas. Observe que o número de capítulos nos *cookbooks* é inferior aos valor do parâmetro K (15) usado no LDA. Isso ocorre porque alguns tópicos não são preenchidos com nenhuma receitas, devido às condições verificadas no algoritmo e também porque os capítulos com menos de três receitas são removidos dos *cookbooks*.

Tabela 4.2: Métricas sobre os *cookbooks* construídos

API	# de Capítulos	# de Receitas
SWT	12	69
LINQ	9	94
QT	12	119

O título de cada capítulo corresponde aos *top-n* termos minerados pelo LDA para o tópico que deu origem ao capítulo. As Tabelas 4.3, 4.4 e 4.5 mostram para as APIS SWT, LINQ e QT, respectivamente, os títulos dos capítulos dos *cookbooks* gerados, usando $n = 5$, o que significa que o título é formado pelos 5 termos mais representativos do tópico. As tabelas também mostram o número de receitas presentes em cada capítulo e três exemplos dos títulos de algumas receitas em cada capítulo. Observe que os títulos dos capítulos contém *stems* ao invés de palavras (e.g., "tabl" ao invés de "table", "arrai" ao invés de "array"), devido ao pré-processamento efetuado antes da aplicação do LDA.

O número de receitas é variável entre os capítulos, principalmente no caso dos *cookbooks* sobre LINQ e QT, em que se verifica que existem capítulos com quantidade de receitas bem superior a outros capítulos. Por exemplo, para o LINQ, o capítulo representado pelo título "list item collect element arrai" possui 43 receitas, e no caso do QT, o capítulo com título "file compil build instal librari" possui 25 receitas. Acredita-se que isso ocorra porque para uma mesma API podem existir temas mais recorrentes que outros. No caso do LINQ, o capítulo "list item collect element arrai" está ligado à manipulação de listas, o que parece ser um dos principais usos do LINQ (essa API é muito utilizada, por exemplo, para encontrar um elemento específico dentro de uma lista, sem ter que percorrê-la). Assim, para uma mesma API, a quantidade de informações sobre os seus temas pode variar no SO, uma vez que os temas mais populares tendem a gerar mais perguntas. Um

estudo feito por Jiau e Yang [30] mostrou que existe uma desigualdade na alocação de recursos no SO, i.e., uma grande quantidade de recursos (e.g., *threads*) está alocada a um percentual pequeno de tópicos, o que de certa forma corrobora a hipótese de que alguns assuntos são mais discutidos que outros no SO.

4.4 Sujeitos Humanos

Os sujeitos humanos que participaram do estudo para avaliar os *cookbooks* são estudantes de pós-graduação em Ciência da Computação (mestrado ou doutorado) e/ou profissionais da área de desenvolvimento de software (algumas pessoas se enquadram em ambas as categorias). Foram convidados para participação no estudo todos os alunos do Programa de Pós-Graduação em Ciência da Computação da UFU e alguns desenvolvedores de software de empresas de Uberlândia.

Tabela 4.3: Exemplos de receitas presentes no *cookbook* sobre SWT

Título do Capítulo	Qtde de Receitas	Exemplo de Títulos de Receitas
print file imag org.eclipse.e4 pdf	6	1- How to display PDFs in a SWT-Application? 2- Zest: export diagram to an image/pdf 3- SWT: How to do High Quality Image Resize
tabl column row cell tableview	8	1- How to know which row has been selected in a TableViewer? 2- Inserting a Button to JFace Table 3- How do I change the index of a TableItem?
imag color draw border font	7	1- Setting Colors in SWT 2- How to draw swt image? 3-Custom widget shapes in SWT
thread ui progress ui_thread call	4	1- Updating SWT objects from another thread 2- How to update a GUI from another thread in Java 3- Detecting when a user is finished resizing SWT shell
text menu kei field press	8	1- How to detect ctrl-f in my SWT application 2- Validation in swt text 3- How to set Cursor position in SWT Text
button event click listen mous	7	1- org.eclipse.swt.widgets.Button click from code 2- SWT event propagation 3- How to dynamically create SWT buttons and their actions?
tree select list item model	4	1- how to Access combo box of a IContributionItem 2- JFace treeViewer select added item 3- SWT: How to find which item is selected?
creat eclips widget add call	5	1- How to create a balloon tool tip for text box in SWT 2- SWT Controls with predefined styles 3- SWT remove all listener from StyledText
file jar eclips icon plugin	5	1- how to make a jar file that include dll files 2- usage of maven tycho-p2-plugin with SWT 3- SWT jar for different platform
wizard dispos page tab file	3	1- Get Active Tab in SWT TabFolder 2- How can I hide a CTabItem in a CTabFolder 3- Text outline in swt
composit size layout scroll resiz	7	1- How to change parent of components with SWT? 2- FormLayout, FormData and controls 3- SWT RowLayout right-to-left?
dialog shell window close screen	5	1- Display parent modal dialog with SWT 2- SWT Java: How to prevent window from resizing? 3- Disable close button in java JFace dialog?

Inicialmente 33 pessoas manifestaram interesse em participar do estudo. Com o objetivo de preparar os participantes para avaliação dos *cookbooks*, foi elaborado um documento de treinamento (Apêndice B.1) que contém:

- Informações a respeito do objetivo do estudo;

Tabela 4.4: Exemplos de receitas presentes no *cookbook* sobre LINQ

Título do Capítulo	Qtde de Receitas	Exemplo de Títulos de Receitas
express lambda dynam lambda_express tree	7	1- Like in Lambda Expression and LINQ 2- How is LINQ compiled into the CIL? 3- How to 'select new' inside Linq lamda expression?
type properti anonym creat anonym_type	6	1- How to convert linq results to HashSet or HashedSet 2- Select Multiple Fields from List in Linq 3- Determine if collection is of type IEnumerable<T>
sql databas entiti tabl data	6	1- Can I return the 'id' field after a LINQ insert? 2- Update using linq 3- LINQ to Entities how to update a record
group column row date tabl	3	1- How to compare only date components from DateTime in EF? 2- How to calculate sum of a DataTable's Column in LINQ (to Dataset)? 3- C# LINQ Query - Group By
queri sql statement select write	13	1- Linq Query to Select Top 5 2- Where IN clause in LINQ 3- LINQ Distinct()
perform loop iter call enumer	8	1- LINQ equivalent of foreach for IEnumerable<t></t> 2- How to replace for-loops with a functional statement in C#? 3- Is yield useful outside of LINQ?
list item collect element arrai	43	1- Find an item in List by LINQ? 2- Compare two lists C# linq 3- linq where list contains any in list
xml file element node attribut	5	1- Ignore namespaces in LINQ to XML 2- XElement namespaces (How to?) 3- Serialize an object to XElement and Deserialize it in memory
tabl join entiti properti relationship	3	1- Convert SQL to Linq left join with null 2- Entity framework left join 3- JOIN and LEFT JOIN equivalent in LINQ

- Definições sobre: *cookbooks*; receitas; as três APIs escolhidas para o estudo; os critérios utilizados para avaliar o coobook (as mesmas informações presentes na Seção 4.2);
- Instruções gerais sobre como proceder durante a avaliação dos *cookbooks*. Dentre essas instruções foi destacada a necessidade de sinceridade durante as avaliações, para que os itens que os avaliadores considerassem ruins fossem avaliados com nota ruim.

Também foi enviado aos participantes um documento (Apêndice B.2) contendo um conjunto de 9 perguntas (8 de múltipla escolha e 1 de "V ou F") que foram elaboradas para assegurar que o participantes estudaram e compreenderam o documento de treinamento. Os participantes devolveram as perguntas respondidas e caso houvesse alguma resposta incorreta, ele eram solicitados a reler o documento de treinamento e a responder novamente às perguntas que eles haviam errado na resposta. Dos 33 voluntários, apenas 23 devolveram as respostas. A maior parte dos participantes acertou todas as respostas na primeira tentativa. Apenas 3 participantes erraram alguma resposta e tiveram que se submeter à releitura do documento de treinamento e a responder novamente alguma pergunta. Desses 23, apenas 16 completaram a avaliação dos *cookbooks* (através do preenchimento de questionários, o que será explicado na Seção 4.5.3). Os nomes desses

Tabela 4.5: Exemplos de receitas presentes no *cookbook* sobre QT

Título do Capítulo	Qtde de Receitas	Exemplo de Títulos de Receitas
string qstring pointer type data	19	1- How to convert int to QString? 2- How to convert QString to std::string? 3- Concatenating two QStrings with an integer
window platform linux mac process	5	1- Detect if stdin is a terminal or pipe in C/C++/Qt? 2- Getting memory information with Qt 3- How to enumerate volumes on Mac OS X?
item model data row view	8	1- Qt How to make a column in QTableWidget read only 2- How can use QTableWidget 3- Qt QTableView how to have a checkbox only column
imag color style background stylesheet	10	1- QLabel: set color of text and background 2- how to convert an opencv cv::Mat to qimage 3- QT QImage, how to extract RGB?
event mous qml click kei	6	1- How can I simulate user interaction (key press event) in Qt? 2- Qt: Defining a custom event type 3- When a Qt widget gets focus
creat http call document find	6	1- Is it possible to use cin with Qt? 2- Naming convention for Qt widgets 3- Do Gui testing in Qt
widget layout size design resiz	9	1- Qt: how to set main window's initial position? 2- Qt Layout on QMainWindow 3- Hiding a QWidget on a QToolbar?
text menu icon line qtextedit	3	1- How to connect menu click with action in Qt Creator? 2- In Qt 4.7, how can a pop-up menu be added to a QToolbar button? 3- How to gray out a menu item in Qt
signal slot thread connect call	7	1- Passing an argument to a slot 2- private/public qt signals 3- Qt:how to give a delay in loop execution
file directori path creat folder	13	1- Get filename from QFile? 2- Copy directory using Qt 3- c++ qt read xml file
window button dialog click close	8	1- How to show another window from mainwindow in QT 2- Yes/No message box using QMessageBox 3- Qt - QDialog's "?"button
file compil build instal librari	25	1- Building multiple targets in Qt / Qmake 2- How do I utilise all the cores for nmake? 3- how to add zlib to an existing qt installation

participantes serão mantidos em sigilo e eles serão referenciados nesta dissertação por "sujeito i", com i variando de 1 até 16.

Com o objetivo de verificar o nível dos participantes em relação às APIs consideradas no estudo, foram feitas as seguintes perguntas a eles:

- Qual seu nível (nenhum, iniciante, intermediário ou avançado) em relação a cada uma das APIs: SWT, LINQ e QT?
- Caso já tenha trabalhado com as APIs citadas, descreva o tipo de projeto e o porte do mesmo (pequeno, médio e grande).

Os resultados para essas perguntas são mostrados nas Tabelas 4.6, 4.7 e 4.8. Observe que a maior parte dos sujeitos nunca trabalhou com as APIs escolhidas: apenas 4 e 1 pessoas se declararam "iniciantes" às APIs SWT e QT, respectivamente, e apenas 1 sujeito se declarou "intermediário" em relação à API LINQ.

Tabela 4.6: Respostas dos sujeitos humanos em relação à API SWT

Sujeito	Nível em Relação à API SWT	Tipo de Projeto no qual usou a API
1	Nenhum	Nenhum
2	Nenhum	Nenhum
3	Iniciante	Fins Acadêmicos
4	Nenhum	Nenhum
5	Nenhum	Nenhum
6	Nenhum	Nenhum
7	Nenhum	Nenhum
8	Nenhum	Nenhum
9	Iniciante	Pequenos Projetos (Protótipos)
10	Nenhum	Nenhum
11	Nenhum	Nenhum
12	Iniciante	Protótipo
13	Iniciante	Protótipo
14	Nenhum	Nenhum
15	Nenhum	Nenhum
16	Nenhum	Nenhum

Tabela 4.7: Respostas dos sujeitos humanos em relação à API LINQ

Sujeito	Nível em Relação à API LINQ	Tipo de Projeto no qual usou a API
1	Nenhum	Nenhum
2	Nenhum	Nenhum
3	Nenhum	Nenhum
4	Nenhum	Nenhum
5	Nenhum	Nenhum
6	Nenhum	Nenhum
7	Nenhum	Nenhum
8	Nenhum	Nenhum
9	Nenhum	Nenhum
10	Nenhum	Nenhum
11	Nenhum	Nenhum
12	Nenhum	Nenhum
13	Nenhum	Nenhum
14	Intermediário	Projetos de pequeno/médio porte. Tanto em projetos criados para testes quanto em projetos já em produção
15	Nenhum	Nenhum
16	Nenhum	Nenhum

Tabela 4.8: Respostas dos sujeitos humanos em relação à API QT

Sujeito	Nível em Relação à API QT	Tipo de Projeto no qual usou a API
1	Nenhum	Nenhum
2	Nenhum	Nenhum
3	Nenhum	Nenhum
4	Nenhum	Nenhum
5	Iniciante	Protótipos
6	Nenhum	Nenhum
7	Nenhum	Nenhum
8	Nenhum	Nenhum
9	Nenhum	Nenhum
10	Nenhum	Nenhum
11	Nenhum	Nenhum
12	Nenhum	Nenhum
13	Nenhum	Nenhum
14	Nenhum	Nenhum
15	Nenhum	Nenhum
16	Nenhum	Nenhum

4.5 Procedimentos

Nessa seção são apresentadas informações relativas ao processo de avaliação dos *cookbooks* pelos participantes do estudo. A Seção 4.5.1 apresenta o processo de construção de versões alteradas dos *cookbooks* (os chamados *Controlled Cookbooks*) que foram criados para controlar a avaliação dos sujeitos humanos a itens sabidamente ruins no que diz respeito às propriedades desejáveis de *cookbooks*. A Seção 4.5.2 trata dos capítulos e receitas que foram selecionados para serem avaliados pelos sujeitos humanos. A Seção 4.5.3 apresenta os questionários por meio dos quais os participantes do estudo avaliaram os *cookbooks*.

4.5.1 Construção dos *Controlled Cookbooks*

O experimento conduzido tem como objetivo apresentar os *cookbooks* gerados aos participantes e solicitar que os mesmos os avaliem por meio dos critérios definidos na Seção 4.2. Além disso, foram elaboradas algumas perguntas discursivas para descobrir qual o papel no aprendizado de APIs os *cookbooks* podem desempenhar.

Para garantir que os participantes estavam sendo honestos durante a avaliação e que eles realmente haviam entendido como avaliar os critérios definidos, foram construídas versões modificadas dos *cookbooks* para as três APIs. Essas versões modificadas são chamadas de *Controlled Cookbooks*. Dessa forma, o conjunto de *cookbooks* considerado neste estudo possui seis elementos: **SWT-*Cookbook***, **SWT-Controlled-*Cookbook***, **LINQ-*Cookbook***, **LINQ-Controlled-*Cookbook***, **QT-*Cookbook*** e **QT-Controlled-*Cookbook***. Os *cookbooks* não-*Controlled* (i.e., **SWT-*Cookbook***, **LINQ-*Cookbook*** e **QT-*Cookbook***) são os construídos pela estratégia proposta nesta dissertação e cujas informações foram mostradas na Seção 4.3.

Os *controlled cookbooks* foram construídos a partir da versão original do *cookbook* para cada API. Assim, o *cookbook* **SWT-Controlled-*Cookbook***[4] foi construído a partir do *cookbook* **SWT-*Cookbook***[5]. O procedimento para construir os *controlled cookbooks* é descrito na sequência e consiste em acrescentar aos *cookbooks* originais (os *cookbooks* de fato produzidos pela estratégia proposta e que foram mostrados na Seção 4.3) "itens" (i.e., receitas e capítulos) sabidamente ruins no que diz respeito aos critérios explicados na Seção 4.2.

Para o critério **Semântica dos Capítulos (*Smt*)** foram criados títulos de capítulos que não remetem a tema nenhum. Em outras palavras, foram escolhidos termos que aparentemente não têm nenhuma relação entre si. No caso do *cookbook* **SWT-Controlled-*Cookbook***, foi criado um capítulo cujo título é representado pelos termos: "pdf press thread eclipse wizard". Dessa forma, o *cookbook* **SWT-Controlled-*Cookbook*** possui 13

[4]http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção swt_fake
[5]http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção swt

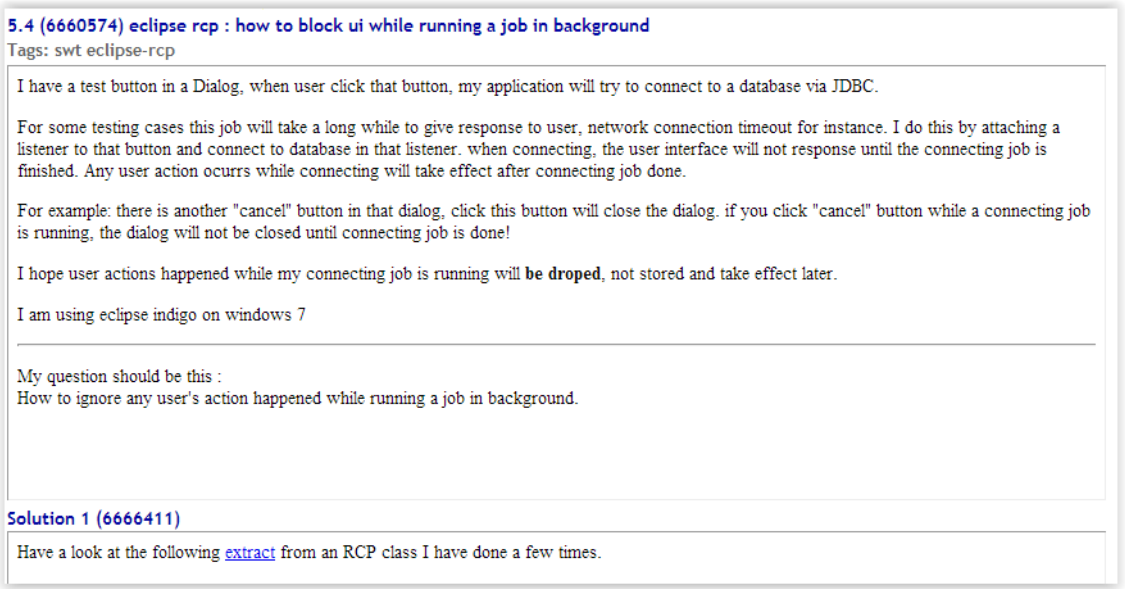
capítulos: os 12 originalmente existentes no *cookbook* **SWT-*Cookbook*** e o capítulo adicional criado. Esse capítulo foi preenchido com três pares de pergunta/resposta quaisquer da API SWT (simplesmente para o capítulo não ficar vazio). De maneira semelhante, para o *cookbook* **LINQ-Controlled-*Cookbook***[6], foi criado um capítulo cujo título é representado pelos termos: "group color xml anonym_type loop". Assim, o *cookbook* **LINQ-Controlled-*Cookbook*** possui 10 capítulos: Os 9 originalmente existentes no *cookbook* **LINQ-*Cookbook***[7] e o capítulo adicional criado. Esse capítulo também foi preenchido com três pares de pergunta/resposta quaisquer da API LINQ. Seguindo a mesma lógica, ao *cookbook* **QT-Controlled-*Cookbook***[8] foi adicionado o capítulo "row imag compil slot event", de maneira que este *cookbook* ficasse com um total de 13 capítulos, uma vez que o *cookbook* original **QT-*Cookbook***[9] tem 12 capítulos.

Em relação ao critério **Relacionamento com o capítulo (*Relac*)** a ideia é inserir um par p de pergunta/resposta em capítulo c, sendo que não existe nenhuma relação entre p e os termos presentes no título de c. Por exemplo, em um capítulo cujos termos remetem ao assunto "array/listas", insere-se um par de pergunta/resposta que trata de "leitura/escrita em arquivos de texto". No caso do **SWT-Controlled-*Cookbook*** foi inserida a receita com título "Add a Key Listener to TitleAreaDialog" no capítulo "tabl column row cell tableview". Dessa forma, este capítulo que no *cookbook* **SWT-*Cookbook*** possui 8 receitas, fica com 9 receitas. Para o *cookbook* **LINQ-Controlled-*Cookbook*** foi introduzida a receita "c# why is the intersect method returning this?" no capítulo "group column row date tabl". Este capítulo ficou 4 receitas, em contraste com o mesmo capítulo no *cookbook* **LINQ-*Cookbook***, que possui 3 receitas. E finalmente, para o *cookbook* **QT-Controlled-*Cookbook***, a receita "Memory management in Qt?" foi inserida no capítulo "event mous qml click kei", que ficou com um total de 7 receitas (6 originais mais 1 *controlled*).

Para o critério **Adequação para ser parte de um *cookbook* (*Adeq*)** foram inseridas nos *cookbooks controlled* pares de pergunta/resposta cujas perguntas não são *How-To-Do-It*, o que os tornam inadequados para serem considerados receitas. No caso do *cookbook* **SWT-Controlled-*Cookbook*** foi incluída uma receita com o título "Eclipse Bug: Unhandled event loop exception No more handles", que é claramente relacionada a um problema em código-fonte (i.e., um *bug*). Para o *cookbook* **LINQ-Controlled-*Cookbook***, foi introduzida uma receita com o título "What's the hardest or most misunderstood aspect of LINQ?", que trata de aspectos conceituas de LINQ e por isso não é um *How-To-Do-It*. No caso do *cookbook* **LINQ-Controlled-*Cookbook*** foi incluída a receita "Problem in threading in QT", que também é um par relacionado a um problema em uma aplicação QT, e sendo assim, não é um *How-To-Do-It*.

[6] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção linq_fake
[7] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção linq
[8] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção qt_fake
[9] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção qt

Para o critério **Auto-contenção (*AutoCont*)**, foram incluídas nos *cookbooks controlled* pares de pergunta/resposta totalmente dependentes do acesso a *links* para que seja possível o entendimento do par. No *cookbook* **SWT-Controlled-*Cookbook*** foi incluído o par mostrado na Figura 4.1. Observe que a resposta deste par apenas contém uma frase que, por meio de um *link* HTML, direciona o usuário para um *site* externo onde ele terá acesso ao exemplo que poderá ajudá-lo. Nos casos dos *cookbooks* **LINQ-Controlled-*Cookbook*** e **QT-Controlled-*Cookbook*** também foram incluídos pares semelhantes ao mostrado para o SWT, i.e., pares cujas respostas apenas apontavam para fontes externas.


5.4 (6660574) eclipse rcp : how to block ui while running a job in background

Tags: swt eclipse-rcp

I have a test button in a Dialog, when user click that button, my application will try to connect to a database via JDBC.

For some testing cases this job will take a long while to give response to user, network connection timeout for instance. I do this by attaching a listener to that button and connect to database in that listener. when connecting, the user interface will not response until the connecting job is finished. Any user action ocurrs while connecting will take effect after connecting job done.

For example: there is another "cancel" button in that dialog, click this button will close the dialog. if you click "cancel" button while a connecting job is running, the dialog will not be closed until connecting job is done!

I hope user actions happened while my connecting job is running will **be droped**, not stored and take effect later.

I am using eclipse indigo on windows 7

My question should be this :
How to ignore any user's action happened while running a job in background.

Solution 1 (6666411)

Have a look at the following extract from an RCP class I have done a few times.

Figura 4.1: SWT - Par de pergunta/resposta com informações não auto-contidas

Para o critério **Reprodutibilidade dos códigos-fontes (*Reprod*)**, foram inseridos nos *cookbooks controlled* pares de pergunta/resposta sem código-fonte ou com trechos de código bastante minimalistas. Nos casos dos *cookbooks* **SWT-Controlled-*Cookbook*** e **LINQ-controlled-*Cookbook*** foram introduzidos pares de pergunta/resposta desprovidos de código. Já para o *cookbook* **QT-Controlled-*Cookbook*** foi incluído um par (Figura 4.2) no qual a resposta contém apenas uma linha de código-fonte na qual um método está sendo chamado, porém não é mostrado como criar o objeto "item" que está invocando o método e também o objeto "color" que é passado como parâmetro ao método sendo chamado. Ou seja, trata-se de um trecho que deverá sofrer modificações (e.g., mostrar a criação dos objetos usados) para que ele possa ser compilado e executado.

Como os itens *controlled* incluídos nos *cookbook* são ruins no que diz respeito aos critérios definidos, espera-se que eles sejam avaliados negativamente (notas -1 ou -2). Isso é uma forma de garantir que os participantes do estudo de fato entenderam como avaliar os itens dos *cookbooks* e que eles não estão "chutando" ou avaliando positivamente apenas para beneficiar os resultados de estudo.

Os participantes do estudo irão avaliar uma amostra dos capítulos e receitas dos *co-*

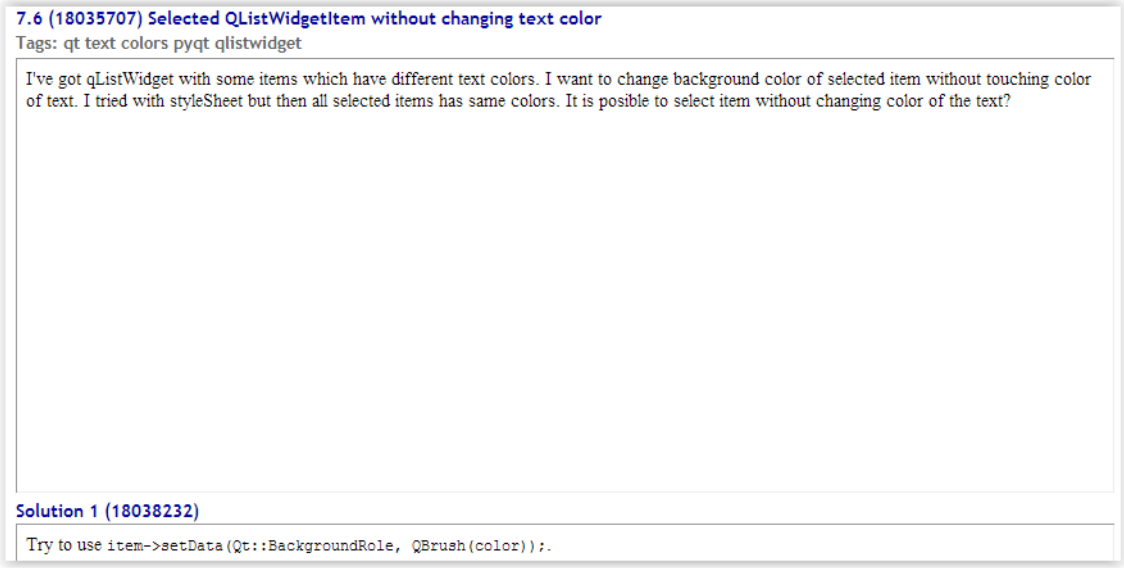


Figura 4.2: Par de pergunta/resposta com código minimalista.

okbooks sobre as três APIs conforme os critérios definidos. Além disso, eles terão que responder a perguntas discursivas com o objetivo de descobrimos o potencial dos *cookbooks* no aprendizado de APIs. Porém, o fato de termos criados *cookbooks* com itens *controlled* (**SWT-Controlled-*Cookbook***, **LINQ-Controlled-*Cookbook***, **QT-Controlled- *Cookbook***), pode influenciá-los nas respostas às perguntas discursivas, uma vez que eles terão que opinar sobre um *cookbook* no qual exitem itens ruins que foram incluídos artificialmente. Por essa razão, os participantes foram sendo incluídos em um de três grupos, à medida que devolviam às respostas relativas ao documento de treinamento. Os grupos criados foram:

- **Grupo-SWT-Controlled**: Os participantes incluídos neste grupo irão avaliar os *cookbooks* **SWT-Controlled-*Cookbook***, **LINQ-*Cookbook*** e **QT-*Cookbook***. Porém, as perguntas discursivas serão apenas sobre os dois *cookbooks* não-*controlled* (**LINQ-*Cookbook*** e **QT-*Cookbook***);
- **Grupo-LINQ-Controlled**: Os participantes incluídos neste grupo irão avaliar os *cookbooks* **SWT-*Cookbook***, **LINQ-Controlled-*Cookbook*** e **QT-*Cookbook***. Porém, as perguntas discursivas serão apenas sobre os dois *cookbooks* não-*controlled* (**SWT-*Cookbook*** e **QT-*Cookbook***);
- **Grupo-QT-Controlled**: Os participantes incluídos neste grupo irão avaliar os *cookbooks* **SWT-*Cookbook***, **LINQ-*Cookbook*** e **QT-Controlled-*Cookbook***. Porém, as perguntas discursivas serão apenas sobre os dois *cookbooks* não-*controlled* (**SWT-*Cookbook*** e **LINQ-*Cookbook***).

A Figura 4.3 ilustra a divisão dos *cookbook* entre os três grupos.
Como os participantes responderão a perguntas discursivas somente sobre os *cookbooks*

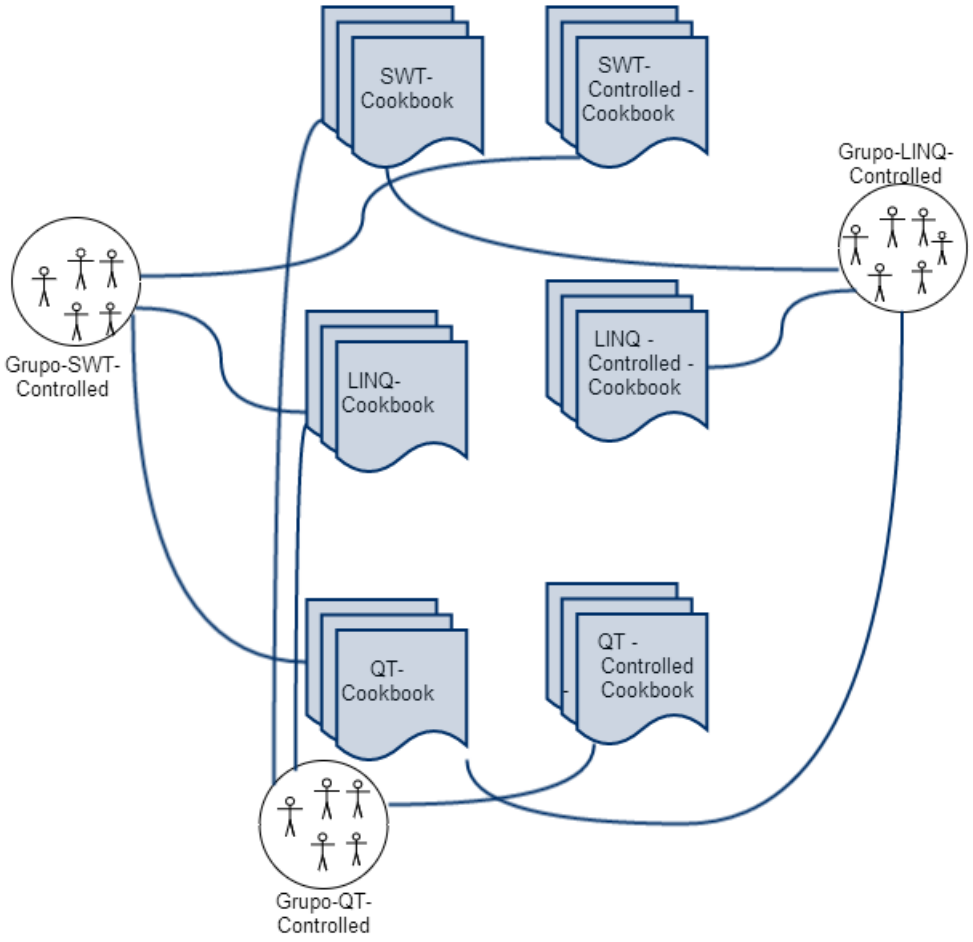


Figura 4.3: Mapeamento dos grupos de sujeitos humanos para os *cookbooks* avaliados.

de fato produzidos pela abordagem apresentada (i.e., sobre os não-*controlled*), espera-se ser possível entender o potencial da abordagem em relação ao aprendizado de APIs.

A divisão dos participantes entre os três grupos foi feita de maneira a balancear o número membros de cada grupo (à medida que eles devolviam as respostas ao documento de treinamento, eles eram incluídos em um dos grupos). A atribuição dos participantes aos grupos ficou assim:

- Grupo-SWT-Controlled: sujeito 4, sujeito 5, sujeito 8, sujeito 10, sujeito 16;
- Grupo-LINQ-Controlled: sujeito 3, sujeito 9, sujeito 11, sujeito 12, sujeito 14, sujeito 15;
- Grupo-QT-Controlled: sujeito 1, sujeito 2, sujeito 6, sujeito 7, sujeito 13.

Dessa forma, dois grupos (Grupo-SWT-Controlled e Grupo-QT-Controlled) ficaram com cinco participantes e um (Grupo-LINQ-Controlled) ficou com seis pessoas.

4.5.2 Amostragem Utilizada na Avaliação

A avaliação de todas as receitas e capítulos dos *cookbooks* tomaria um tempo demasiadamente grande dos participantes devido ao tamanho dos mesmos. Por essa razão, os participantes avaliaram uma amostra dos capítulos e receitas dos *cookbooks*.

Observe que dos cinco critérios definidos, um é referente a capítulos (*Smt*) e quatro são aplicáveis a receitas (*Relac*, *Adeq*, *AutoCont* e *Reprod*). Por isso, os participantes inicialmente irão avaliar um conjunto de capítulos e dar nota ao critério *Smt*. Em uma segunda etapa, eles avaliarão receitas para dar nota aos critérios *Relac*, *Adeq*, *AutoCont* e *Reprod*.

A amostragem das receitas é feita de maneira simultânea para o versão original e *controlled* dos coobooks para uma API. Assim, considere que esteja sendo feita a amostragem das receitas para os *cookbooks* **SWT-*Cookbook*** e **SWT-Controlled-*Cookbook***:

1. Sorteia-se uma receita de cada capítulo do *cookbook* **SWT-*Cookbook***. A ideia é selecionar receitas de todos os capítulos para que a amostra represente melhor o *cookbook* como um todo. No caso do *cookbook* **SWT-*Cookbook***, como ele possui 12 capítulos, são sorteadas 12 receitas. As receitas sorteadas farão parte tanto da avaliação do *cookbook* **SWT-*Cookbook***, quanto do **SWT-Controlled-*Cookbook***. Observe que o *cookbook* **SWT-Controlled-*Cookbook*** possui todas essas receitas não-*controlled*, pois ele é construído através da inclusão de receitas *controlled* ao *cookbook* **SWT-*Cookbook***;
2. No caso do **SWT-Controlled-*Cookbook***, além das 12 receitas sorteadas no passo 1, são selecionadas as quatro receitas *controlled* incluídas artificialmente neste *cookbook*. Lembre-se que durante a construção deste *cookbook*, foram incluídas quatro receitas escolhidas "a dedo": uma que é ruim em relação ao critério *Relac*, uma que é ruim em relação ao critério *Adeq*, uma que é ruim em relação ao critério *AutoCont* e uma que é ruim em relação ao critério *Reprod*. Assim, para a avaliação do *cookbook* **SWT-Controlled-*Cookbook***, foram selecionadas 16 (12 + 4) receitas para serem avaliadas pelo público;
3. No caso do **SWT-*Cookbook***, além das 12 receitas sorteadas no passo 1, são selecionadas aleatoriamente mais quatro receitas. Isso foi feito para que todos os três grupos de participantes avaliem 16 receitas relacionadas à API SWT.

A seleção de receitas para as APIs LINQ e QT segue o mesmo padrão da apresentada para o SWT.

A amostragem das capítulos também é feita de maneira simultânea para o versão original e *controlled* dos coobooks para uma API. Assim, considere que esteja sendo feita a amostragem dos capítulos para os *cookbooks* **SWT-*Cookbook*** e **SWT-Controlled-*Cookbook***:

1. Ordena-se os capítulos do *cookbook* **SWT-*Cookbook*** de maneira ascendente pelo número de receitas (i.e., o primeiro capítulo depois desta ordenação é o "wizard dispos page tab file" que possui apenas três receitas, e o último é o capítulo "tabl column row cell tableview" ou "text menu kei field press" que possuem oito receitas);
2. Identifica-se o capítulo central (caso o número de capítulos seja ímpar) ou os dois capítulos centrais (caso o número de capítulos seja par) desta lista (similar ao cálculo da mediana). São sorteadas quatro capítulos: dois com menos receitas do que o(s) capítulo(s) central(is) e dois com mais. No caso do *cookbook* **SWT-*Cookbook*** que possui um número par de receitas (12) e os dois capítulos centrais são o "dialog shell window close screen", com 5 receitas e o "print file imag org.eclipse.e4 pdf", com 6 receitas, foram sorteados os capítulos "wizard dispos page tab file" (3 receitas), "file jar eclips icon plugin" (5 receitas), "imag color draw border font" (7 receitas) e tabl column row cell tableview" (8 receitas). A justificativa para essa abordagem é selecionar capítulos de tamanhos variados para a amostra melhor representar o *cookbook* como um todo. As receitas sorteadas farão parte tanto da avaliação do *cookbook* **SWT-*Cookbook***, quanto do **SWT-Controlled-*Cookbook***. Observe que o *cookbook* **SWT-Controlled-*Cookbook*** possui todas os capítulos presentes no *cookbook* **SWT-*Cookbook*** e mais um que foi artificialmente criado;
3. No caso do **SWT-Controlled-*Cookbook***, além dos quatro capítulos sorteadas no passo 1, é selecionado o capítulo *controlled* ("pdf press thread eclipse wizard") que foi artificialmente incluído neste *cookbook*. Assim, o grupo alocado ao *cookbook* **SWT-Controlled-*Cookbook*** irá avaliar cinco capítulos sobre a API SWT (os quatro selecionados no passo 2 e o capítulo *controlled*) e os grupos alocados ao *cookbook* **SWT-*Cookbook*** irão avaliar os quatro capítulos selecionados no passo 2.

A lista completa de receitas e capítulos selecionados pela amostragem está no Apêndice C.1.

4.5.3 Questionários para Avaliação dos *Cookbooks*

Os *cookbooks* foram avaliados pelos participantes do estudo através do preenchimentos de questionários. Foram preparados três tipos de questionários: uma para cada um dos três grupos de participantes (Grupo-SWT-Controlled, Grupo-LINQ-Controlled e Grupo-SWT-Controlled). O questionário do grupo Grupo-SWT-Controlled contém perguntas a respeito dos *cookbooks* **SWT-Controlled-*Cookbook***, **LINQ-*Cookbook*** e **QT-*Cookbook***. O questionário do grupo Grupo-LINQ-Controlled contém perguntas a respeito dos *cookbooks* **SWT-*Cookbook***, **LINQ-Controlled-*Cookbook*** e **QT-*Cookbook***. Por fim, o questionário do grupo Grupo-QT-Controlled contém perguntas a respeito dos *cookbooks* **SWT-*Cookbook***, **LINQ-*Cookbook*** e **QT-Controlled-*Cookbook***. Apesar

de terem sido construídos *cookbooks controlled*, os avaliadores não tinham ciência deste fato. Assim, ao avaliarem o *cookbook* **SWT-Controlled-*Cookbook***, eles não sabiam que existiam alguns itens (capítulos/receitas) ruins que foram propositalmente incluído nos *cookbook*. Além disso, o nome **SWT-Controlled-*Cookbook*** não foi revelado: eles apenas sabiam que se tratava de um *cookbook* sobre o SWT.

Os questionários foram preparados e hospedados utilizando o serviço LimeSurvey[10]. Os participantes podiam acessar o questionário através de uma URL fornecido pelo serviço. Também foi fornecida aos participantes uma URL para acesso à aplicação *web* construída para visualização dos *cookbooks*. O serviço LimeSurvey permite que o usuários salvem o progresso do preenchimento do questionário e o retomem depois, i.e., não era preciso que eles preenchessem tudo de uma vez. Foi dado um prazo de duas semanas para que os participantes terminassem o preenchimento do questionário.

Cada questionário é composto por três seções. A primeira seção consiste de perguntas para avaliar o critério *Smt* (i.e., essa seção está relacionada às perguntas de pesquisa **RQ1** e **RQ2** do estudo). Para cada capítulo selecionado pelo processo de amostragem, existem duas perguntas no questionário. Uma para avaliar o capítulo em questão quanto ao critério *Smt* e outra para escrever uma frase em linguagem natural que descreva o título do capítulo de uma maneira mais alto nível, caso o avaliador tenha dada nota 2 ou 1 em relação ao critério *Smt* (i.e., para caso ele tenha conseguido perceber algum tema para o capítulo a partir da leitura dos cinco termos representativos). A Figura 4.4 exemplifica essas duas perguntas para um capítulo.

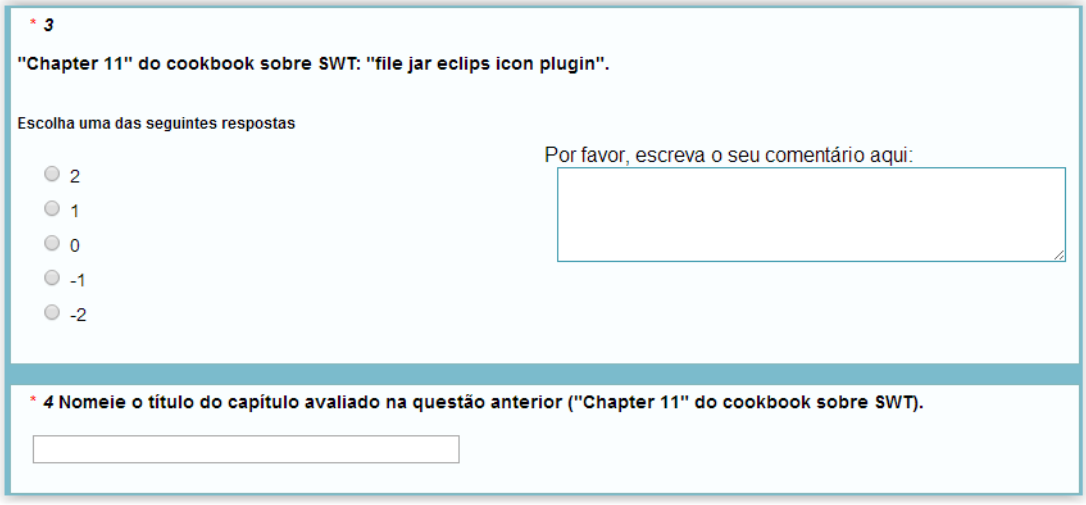


Figura 4.4: Exemplo de perguntas para um capítulo

A segunda parte do questionário consiste de perguntas para avaliar os critérios relativos a receitas (*Relac*, *Adeq*, *AutoCont* e *Reprod*). Dessa forma, este segunda parte está relacionada às perguntas de pesquisa **RQ3**, **RQ4**, **RQ5**, **RQ6**, **RQ7** e **RQ8**. Para

[10]http://www.limesurvey.org/pt/

cada receita, foram criadas quatro perguntas no questionário: uma para cada um destes critérios. Um exemplo é mostrado na Figura 4.5.

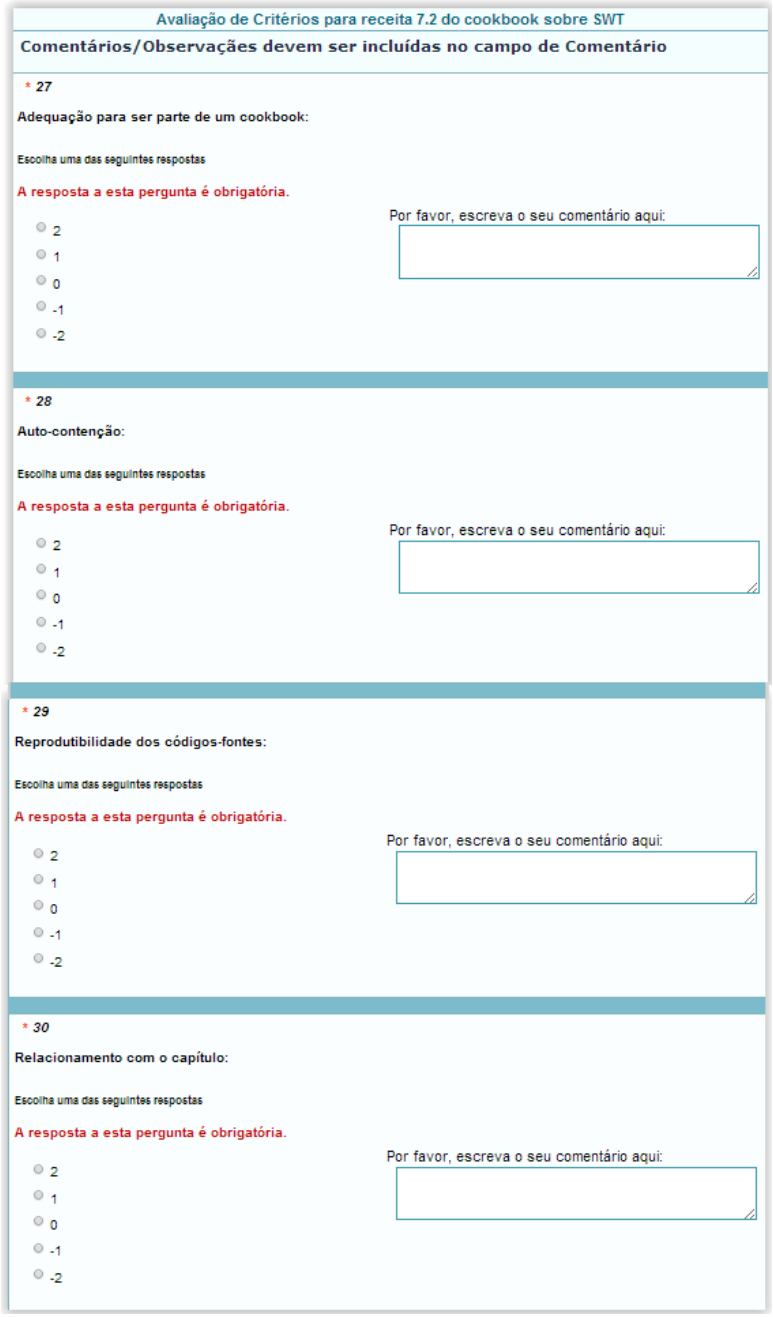


Figura 4.5: Exemplo de perguntas para uma receita

A terceira parte do questionário diz respeito a um conjunto de perguntas (por API) composto por três partes:

- Um pergunta feita para obter as impressões gerais do avaliador sobre o *cookbook* bem como os pontos positivos, pontos negativos e oportunidades de melhoria que ele percebeu (Figura 4.6-a). Esta pergunta está relacionada à pegunta de pesquisa **RQ9**;

- Uma pergunta sobre a utilidade do *cookbook* em relação ao aprendizado da API alvo do *cookbook* em questão (Figura 4.6-b). Esta pergunta está relacionada à pergunta de pesquisa **RQ10** do estudo;
- Uma pergunta para o avaliador dar uma nota geral para o *cookbook* (variando de 0(pior) até 10(melhor)) (Figura 4.6-c).

O grupo Grupo-SWT-Controlled respondeu a essa terceira parte do questionário apenas em relação às APIs LINQ e QT. Similarmente os avaliadores do grupo Grupo-LINQ-Controlled tiveram que responder a essas perguntas apenas para as APIs SWT e QT. Por fim, os participantes do grupo Grupo-QT-Controlled responderem a essas perguntas apenas nos casos das APIs SWT e LINQ.

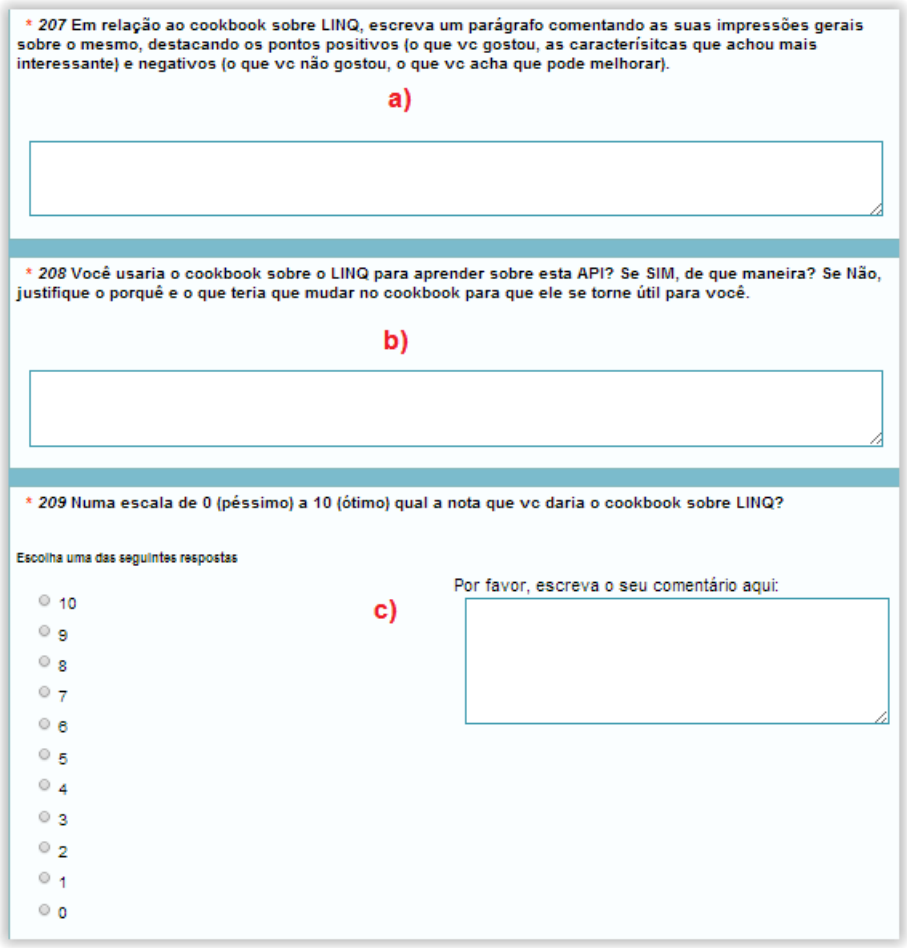


Figura 4.6: Terceiro grupo de perguntas do questionário.

Observe que em todas as perguntas de múltipla escolha do questionário, existem um campo de texto destinado à comentários de avaliador.

Por fim, no final do questionário, um campo de texto foi destinado a comentários adicionais que se fizessem necessários.

Capítulo 5

Resultados e Discussão

Nesse capítulo são apresentados e discutidos os resultados da avaliação feita pelos sujeitos humanos (da Seção 5.1 até a 5.11), bem como são apresentadas as ameaças à validade (Seção 5.12).

5.1 Descarte do Critério *Reprod*

Apesar de ter sido definido o critério *Reprod* para verificar se as receitas atendem ao critério de reprodutibilidade, resolveu-se não considerá-lo. A razão para isso se deve ao fato de que a maioria dos participantes possui nenhum ou pouco conhecimento sobre as APIs estudadas, o que torna difícil o julgamento acerca das modificações que um trecho de código demandaria para ele poder ser compilado/executado. Essa preocupação foi inclusive corroborada pelos comentários feitos por alguns participantes. Por exemplo, um dos avaliadores fez o seguinte comentário: "*Durante a avaliação, eu pude perceber que e difícil avaliar o critério de Reprodutibilidade de código-fonte em uma API e linguagem que você não é familiar. Como você sabe se um código em Qt é compilável sem você ter um mínimo de background em C++ e Qt??*". Por essa razão, os resultados para este critério não serão mostrados no restante do capítulo e não será possível verificar se as receitas atendem ao critério de reprodutibilidade, ou seja, a pergunta de pesquisa **RQ6** não será respondida. A pergunta de pesquisa **RQ7** deverá ser modificada, pois inicialmente ela tem o objetivo de verificar em que extensão as opiniões dos avaliadores se assemelham em relação a cinco critérios (incluindo a reprodutibilidade). Dessa forma, deriva-se uma nova pergunta de pesquisa **RQ7'** que não leva em conta o critério reprodutibilidade. **RQ7': Em que extensão as opiniões dos sujeitos humanos em relação à propriedades "semântica do capítulo", "relacionamento com tema do capítulo", "adequabilidade para ser parte de *cookbook*" e "auto-contenção" se assemelham?** Da mesma forma, a pergunta de pesquisa **RQ8** também deverá ser modificada, pois inicialmente ela possui o objetivo de verificar a quantidade de receitas que atendem a quatro critérios (incluindo a reprodutibilidade). Assim, a partir de **RQ8** deriva-se uma nova per-

gunta de pesquisa **RQ8'** que desconsidera o critério reprodutibilidade. **RQ8': Qual a quantidade de receitas que atendem simultaneamente às propriedades "relacionamento com tema do capítulo", "adequabilidade para ser parte de *cookbook*" e "auto-contenção"?**

Considerações sobre a Avaliação do Critério *Reprod*
Os resultados das avaliações ao critério *Reprod* foram descartadas devido à pequena ou nenhuma experiência dos sujeitos humanos em relação às APIs consideradas no estudo. Por isso, a perguntas **RQ6** foi descartada e as perguntas **RQ7** e **RQ8** foram modificadas, originando as perguntas **RQ7'** e **RQ8'**.

5.2 Respostas aos Itens *Controlled*

O primeiro passo na análise das respostas aos questionário foi em relação às receitas e capítulos *controlled*. Cada participante do estudo avaliou quatro receitas *controlled* e um capítulo *controlled*. Conforme explicado, cada uma das receitas *controlled* foi escolhida por não satisfazer um dentre os critérios *Relac*, *Adeq*, *AutoCont* e *Reprod*. Assim, no caso de uma receita *controlled* que tenha sido escolhida por ser ruim em relação ao critério *Relac*, nos interessa apenas o valor dado a este critério e os valores dados aos outros três critérios serão desconsiderados. Conforme explicado, não serão mostrados os valores relativos ao critério *Reprod*.

As respostas dadas às perguntas relativas aos itens *controlled* são mostrados nas Tabelas 5.1, 5.2 e 5.3 para os três grupos de participantes. Nestas tabelas, as notas com valores positivos foram coloridas de verde (claro no caso de 1 e escuro para 2), as notas com valores negativos foram coloridas de vermelho (claro para -1 e escuro para -2) e as notas com valor igual a 0 não foram coloridas. O esperado é que todas as células das tabelas estivessem pintadas de vermelho, pois os itens *controlled* são ruins quanto aos critérios selecionados. De fato, a maior parte das células das tabelas está em vermelho, o que significa que os itens *controlled*, em sua maioria, foram considerados ruins pelos avaliadores. No caso do SWT, 100% dos itens *controlled* foram avaliados com -1 ou -2 por todos os avaliadores. No caso da API LINQ, apesar de percebermos a predominância do vermelho na tabela (17 de 24), tiveram 6 casos em que os avaliadores não souberam opinar (nota 0) e dois casos avaliados positivamente. Para o QT, também se verifica a predominância do vermelho (12 de 20). Porém, nesta API, os sujeitos 1 e 2 se destacaram. O sujeito 1 deu nota negativa apenas para o critério *Relac*, tendo sido neutro nos demais critérios. Ao ler os comentários feitos por este avaliador, percebe-se que ele não se sentiu seguro na avaliação das APIs LINQ e QT, visto que das três APIs do estudo, ele tinha alguma experiência apenas com o SWT. O comentário feito por ele foi: "*No meu caso, eu só utilizei SWT. Nas questões de múltipla escolha, muitas vezes fiquei com dúvida em*

responder sobre as receitas dos cookbooks sobre LINQ e QT. Em alguns casos, eu compreendi a lógica do cenário e da solução, e então consegui responder algumas perguntas com certeza, mas a maioria, quando era exigido implicitamente um conhecimento prévio do assunto, eu fiquei em dúvida.". No caso do sujeito 2, acredita-se que ele simplesmente "chutou" as respostas do questionário. Observe que dos quatro itens *controlled*, dois ele avaliou positivamente e nos outros dois foi neutro. Outro fato que nos leva a acreditar que o mesmo não levou a sério o preenchimento do questionário, é que na avaliação do critério *Smt*, nos casos em que ele deu nota 2 ou 1 e foi solicitado a escrever uma frase em linguagem natural que descrevesse o tema que ele percebeu a partir dos termos representativos do capítulo, ele simplesmente copiou os cinco termos representativos do capítulo e os colou na resposta (i.e., um simples "ctrl c + ctrl v").

Tabela 5.1: Respostas aos itens *controlled* - Grupo SWT controlled

Sujeito	*Smt*	*Adeq*	*AutoCont*	*Relac*
4	-2	-1	-2	-2
5	-2	-2	-2	-2
8	-1	-2	-2	-2
10	-2	-2	-2	-2
16	-2	-2	-2	-1

Tabela 5.2: Respostas aos itens *controlled* - Grupo LINQ controlled

Sujeito	*Smt*	*Adeq*	*AutoCont*	*Relac*
3	0	2	-1	1
9	-2	-1	0	0
11	-2	-2	-2	-2
12	0	0	-2	0
14	-1	-2	-2	-1
15	-2	-2	-2	-2

Tabela 5.3: Respostas aos itens *controlled* - Grupo QT controlled

Sujeito	*Smt*	*Adeq*	*AutoCont*	*Relac*
1	0	0	0	-2
2	0	1	0	1
6	-1	-2	-1	-1
7	-1	1	-2	-1
13	-2	-1	-1	-1

Para cada usuário, os resultados apresentados durante o restante deste capítulo, dizem respeito somente aos critérios que ele avaliou negativamente na análise dos itens *controlled*. Isso é uma forma de garantir que ele compreendeu como avaliar cada critério cujos resultados serão mostrados ao longo do capítulo. Assim, por exemplo, para o sujeito 4, serão consideradas as notas dadas aos quatro critérios; para o sujeito 9 serão consideradas

as notas relativas aos critérios *Smt* e *Adeq*; para o sujeito 1, apenas serão considerada as notas relativas ao critério *Relac*; e nenhuma das notas dadas pelo sujeito 2 será considerada (i.e., o mesmo foi excluído completamente dos resultados apresentados no decorrer do capítulo).

5.3 Respostas ao Critério *Smt*

A Tabela 5.4 mostra o número de capítulos que cada um dos participantes avaliou em cada um dos cinco valores (2, 1, 0, -1, -2) possíveis para o critério *Smt*, desconsiderando o capítulo *controlled* de cada questionário. Esta tabela também mostra a quantidade de capítulos avaliados com nota positiva (2 ou 1), visto que isso é um informação importante, pois corresponde aos casos em que o participante conseguiu encontrar um significado para o título do capítulo, a partir da leitura dos cinco termos-chave que o constituem. Na Figura 5.1 é mostrada informação semelhante à da tabela. Percebe-se que as barras nas cores verde escuro e claro, que representam as notas 2 e 1 respectivamente, costumam ser maiores do que as outras barras, o que indica que as avaliações foram, em sua maioria, positivas.

Tabela 5.4: Número de capítulos avaliados por valor de *Smt*

Sujeito	2	1	2 ou 1	0	-1	-2
4	5 (41,67%)	0 (0%)	5 (41,67%)	1 (8,33%)	5 (41,67%)	1 (8,33%)
5	7 (58,33%)	1 (8,33%)	8 (66,67%)	1 (8,33%)	0 (0%)	3 (25%)
6	3 (25%)	7 (58,33%)	10 (83,33%)	1 (8,33%)	1 (8,33%)	0 (0%)
7	7 (58,33%)	1 (8,33%)	8 (66,67%)	2 (16,67%)	2 (16,67%)	0 (0%)
8	6 (50%)	3 (25%)	9 (75%)	1 (8,33%)	2 (16,67%)	0 (0%)
9	0 (0%)	4 (33,33%)	4 (33,33%)	1 (8,33%)	1 (8,33%)	6 (50%)
10	2 (16,67%)	3 (25%)	5 (41,67%)	5 (41,67%)	2 (16,67%)	0 (0%)
11	6 (50%)	3 (25%)	9 (75%)	3 (25%)	0 (0%)	0 (0%)
13	4 (33,33%)	5 (41,67%)	9 (75%)	2 (16,67%)	1 (8,33%)	0 (0%)
14	0 (0%)	5 (41,67%)	5 (41,67%)	4 (33,33%)	1 (8,33%)	2 (16,67%)
15	5 (41,67%)	1 (8,33%)	6 (50%)	0 (0%)	3 (25%)	3 (25%)
16	6 (50%)	2 (16,67%)	8 (66,67%)	1 (8,33%)	3 (25%)	0 (0%)

Dos 12 avaliadores considerados na Tabela 5.4, 8 avaliaram pelo menos 6 dos 12 capítulos (i.e., 50%) com nota 1 ou 2. O número médio de capítulos avaliados com nota 1 ou 2, dentre os 12 participantes é de 7,17 (59,75% dos 12 capítulos). Considerando a totalidade de avaliações feitas pelos participantes (i.e., 144 avaliações, pois cada um dos 12 avaliadores avaliou 12 capítulos), foram 51 (35,42%) avaliações com nota 2, 35 (24,31%) com nota 1, 22 (15,28%) com nota 0, 21 (14,58%) com nota -1 e 15 (10,42%) com nota -2. Assim, pode-se dizer que a propriedade de semântica dos capítulos é atendida em grande parte dos *cookbooks*, pois para uma porção considerável dos capítulos foram encontrados

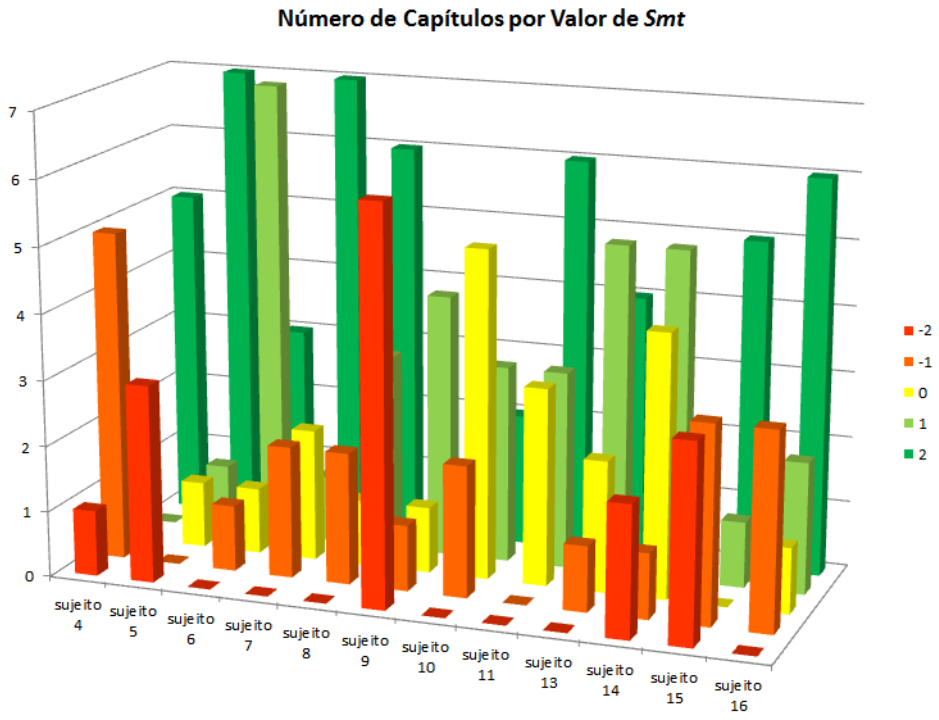


Figura 5.1: Distribuição das notas dos participantes em relação ao critério *Smt*

temas pelos avaliadores. Considerando as avaliações feitas por API (i.e., 48 avaliações, pois cada um dos 12 participantes avaliou quatro capítulos por API), os resultados são:

- SWT: 17 (35,42%) avaliações com nota 2, 16 (33,33%) com nota 1, 3 (6,25%) com nota 0, 8 (16,67%) com nota -1 e 4 (8,33%) com nota -2;
- LINQ: 20 (41,67%) avaliações com nota 2, 9 (18,75%) com nota 1, 7 (14,58%) com nota 0, 8 (16,67%) com nota -1 e 4 (8,33%) com nota -2;
- QT: 14 (29,17%) avaliações com nota 2, 10 (20,83%) com nota 1, 12 (25%) com nota 0, 6 (12,5%) com nota -1 e 6 (12,5%) com nota -2.

Das três APIs, o QT foi a que recebeu menos avaliações com nota 1 ou 2 (24 contra 33 para SWT e 29 para LINQ). A diferença entre as avaliações dos participantes, além da subjetividade inerente, pode ser explicada pelos diferentes *backgrounds* que os participantes têm em relação a programação. Os termos presentes nos títulos dos capítulos são termos técnicos de programação. Assim, acredita-se que a experiência do avaliador com programação e com a API em questão, pode ser um fator que faço-o conhecer mais ou menos a nomenclatura usada no desenvolvimento de software e em especial naquela API, bem como pode influenciar na capacidade do avaliador em relacionar estes termos.

As Tabelas 5.5, 5.6 e 5.7 mostram, para as APIS SWT, LINQ e QT, respectivamente, os títulos atribuídos pelos avaliadores aos cinco termos representativos de cada capítulo selecionado pelo processo de amostragem. A maior parte dos avaliadores escreveu o tema encontrado em inglês. Por isso, os temas que foram escritos em português, foram traduzidos para o inglês, para que tudo ficasse no mesmo idioma. Percebe-se que o número de

participantes que conseguiram atribuir título, varia conforme o capítulo. Por exemplo, 11 participantes atribuíram título ao capítulo "imag color draw border font" do SWT, 2 atribuíram título ao capítulo "wizard dispos page tab file" também do SWT, e ninguém foi capaz de atribuir título ao capítulo "creat http call document find" do QT. Assim, pode-se dizer que os temas de alguns capítulos são mais bem definidos que outros, pois as pessoas tiveram mais facilidade para encontrar temas para alguns capítulos do que para outros.

Em relação ao tema encontrado pelos vários avaliadores para um mesmo capítulo, percebe-se que eles são bastante semelhantes. Por exemplo, para o capítulo "tabl column row cell tableview" do SWT, todos os temas atribuídos pelos participantes estão relacionados a manipulação de tabelas em SWT. Mesmo quando os temas encontrados não possuem exatamente o mesmo sentido, em geral, eles possuem relação entre si. Por exemplo, o capítulo "imag color draw border font" do SWT, recebeu alguns temas como "Working with Colors and Images", "SWT Pallete" e "2D". Apesar de não serem exatamente o mesmo tema, eles possuem forte relação entre si: uma paleta (*Pallete*) de cores em SWT pode ser usada para mudar a cor de imagens, que por sua vez, são elementos de duas dimensões (2D). Porém, percebe-se que existem alguns temas que são ruídos. Por exemplo, para o capítulo "type properti anonym creat anonym_type" do LINQ, dos quatro avaliadores que atribuíram tema, três deles relacionaram o capítulo com o assunto "tipos de dados", e um deles atribuiu um tema que se distanciou bastante: "Manipulation of Queries". Em geral, dentre as pessoas que conseguem atribuir temas a um capítulo, os temas tendem a ter semânticas parecidas.

5.3.1 Análise dos Comentários

Foi feita uma leitura dos comentários que os participantes fizeram em relação às avaliações ao critério *Smt*. Os pontos mencionados foram:

- O uso de *stems* ao invés de palavras "completas" (e.g., "eclips" ao invés de "Eclipse", "tabl" no lugar de "table") não agradou aos avaliadores ou dificultou a análise. Por exemplo, um dos avaliadores disse: "*O termo eclips eu não sei o que significa. Os demais estão relacionados a tipos de arquivos de pacote.*" Outro comentou: "*Não seria image?*" (se referindo ao termo "imag" presente no capítulo "imag color draw border font"). O processo do *stemming* é parte fundamental do pré-processamento aplicado antes da execução do LDA, ou seja, não é aconselhável aplicar o LDA sem ter sido feito o *stemming* dos termos das *threads*. Para resolver este impasse, os *stems* presentes nos títulos dos tópicos minerados pelo LDA poderiam ser convertidos em palavras originais. A dificuldade com essa abordagem é que para um mesmo *stem* podem existar várias palavras (e.g., as palavras "enumerate" e "enumeration" possuem como stems o termo "enumer"), o que exigiria que fosse escolhida apenas

Tabela 5.5: SWT - Títulos atribuído pelos avaliadores em casos de nota 1 ou 2 para o critério *Smt*

Título do Capítulo	Títulos Atribuídos pelos Avaliadores
imag color draw border font	Drawing images and text on the screen Image drawing Working with Colors and Images SWT Palette Drawing images and customizing graphical attributes SWT Graphics Customizing controls Interface elements (front-end) 2D Styling Image Configuration Display elements appearance
wizard dispos page tab file	SWT Tabs Manupulating Tabs
file jar eclips icon plugin	Jar files and plugins Manipulation of .Jar Files Creating jar files with eclipse plugin Creating jar files Executable File Packages and Plugins File, Jars and Eclipse plugins
tabl column row cell tableview	Tables Managing tables in SWT Manipulation of Tables SWT Table Using tables SWT Table Customizing Tables Customizing Tables Tables Tables Table Viewer Configuration Tables and TableViews

uma dessa palavras para ser apresentada no lugar do *stem*. Uma solução seria simplesmente escolher de maneira aleatória uma das palavras;

- Em alguns casos, a avaliador se sentiu inseguro e preferiu não atribuir um tema ao título do capítulo, porém escreveu no campo destinado à comentários um tema que se assemelha ao tema atribuído pelos participantes que encontram um tema. Por exemplo, um avaliador disse "*Não entendi, mas acho que é algo relacionados a tipos de dados*" se referindo ao capítulo "type properti anonym creat anonym_type" do LINQ. Apesar de o usuário ter preferido não atribuir um tema, percebe-se que ele encontrou uma semântica parecida à encontrada por outros avaliadores;
- Um avaliador não deu nota positiva ao critério *Smt* para alguns capítulos, porque achou que os termos do título do capítulo poderiam estar melhor organizados. Ele disse: "*Eu acho que ficaria melhor assim: XML: file, element, node, attribut etc.*" se referindo ao capítulo "xml file element node attribut". Os títulos dos capítulos encontrados pelo LDA são termos "soltos" e não frases completas, por isso a organização/ordenação dos mesmos nem sempre vai agradar os usuário de um *cookbook* produzido pela abordagem proposta nesta dissertação. Uma maneira de resolver essa situação é substituir o título de cada capítulo do *cookbook* por uma frase em linguagem natural que reflita o significado do capítulo, antes de liberar o *cookbook*

Tabela 5.6: LINQ - Títulos atribuído pelos avaliadores em casos de nota 1 ou 2 para o critério *Smt*

Título do Capítulo	Títulos Atribuídos pelos Avaliadores
xml file element node attribut	Working with XML Dealing with xml node elements Querying with LINQ Working with xml documents Working with XML in LINQ Properties in XML XML Xml treatments.
type properti anonym creat anonym_type	Creating anonymous types Manipulation of Queries Working with anonym types Types and Properties
perform loop iter call enumer	How LINQ works and performance Iterating through loops Working with Repetition Structures Iterating over enumerations Iteration, loops and lists Performance in loops Learning to iterate Repetition structures (or loops if you want)
list item collect element arrai	Lists and collections Managing lists Manipulation of Array and Lists Working with collections, lists and arrays Operations on lists and arrays Collections Vectors and data collections Array operations Collections

para o público;

RQ1: Em que extensão os capítulos dos *crowd cookbooks* possuem sentido definido?
Para a maior parte dos capítulos dos *cookbooks* foram encontrados temas (média de 59,75%). **Assim, a resposta à pergunta de pesquisa RQ1 é: a propriedade de semântica dos capítulos é atendida em grande parte dos *crowd cookbooks*.** O número de participantes que conseguiram atribuir significado a cada capítulo varia, indicando que alguns capítulos tem semântica melhor definida que outros. Em geral as semânticas encontradas pelos avaliadores são semelhantes entre si. Como melhorias em relação à semântica dos capítulos, destacam-se a necessidade de formar o títulos dos capítulos com palavras originais ao invés de *stems* e evitar a geração de capítulos com mais de um tema.

- Para alguns capítulos os usuários encontraram mais de um tema e por isso não deram nota positiva ao critério *Smt*. Por exemplo, em relação ao capítulo "creat http call document find", um usuário disse: "*A interpretação que tive foi de dois temas distintos: 'Creating http calls' e 'Finding terms in documents'*". Esse é um problema ligado ao parâmetro K do LDA, i.e, à granularidade dos tópicos encontrados pelo LDA. Talvez se fosse usado um valor maior de K (foi usado o valor 15), esses dois

temas aparecessem como tópicos distintos. Essa é uma questão difícil de resolver, pois conforme explicado, não existe um único valor de K que seja adequado a todos os *datasets*. Porém, como as avaliações ao critério *Smt* tiveram valores razoáveis, mesmo que ainda exista margem para melhoria, acredita-se que o valor de K usado no LDA foi aceitável.

5.4 Sobreposição de Temas entre Capítulos

Em um *cookbook*, cada capítulo deve tratar de um tema específico. Outra característica desejada de um *cookbook* é que não exista a sobreposição de um mesmo assunto ao longo de mais de um capítulo. Para analisar a sobreposição, foi calculado para cada capítulo, o número de termos representativos de seu título que também fazem parte do títulos de outros capítulos. Lembrando que o título de cada capítulo é composto por cinco termos (*stems*). Como o tema de um capítulo pode ser derivado da leitura de seu título, a presença de termos repetidos entre capítulos pode ser um indício da existência de um mesmo tema que está sendo tratado em mais de um capítulo. As informações relativas à sobreposição de termos entre capítulos nos *cookbooks* **SWT-*Cookbook***, **LINQ-*Cookbook*** e **QT-*Cookbook*** são mostradas na Tabela 5.8. Os termos repetidos entre os pares de capítulos são mostrados em negrito.

Tabela 5.7: QT - Títulos atribuído pelos avaliadores em casos de nota 1 ou 2 para o critério *Smt*

Título do Capítulo	Títulos Atribuídos pelos Avaliadores
window platform linux mac process	QT platforms Manipulation of Process on Different Operationg Systems Creating windows for multiple platforms Cross-platform compatibility Differences between platforms Platforms/OS Environment
creat http call document find	—
string qstring pointer type data	Basic types of QT Dealing with QString type data Working with QStrings QString in QT Types of data in QT Variables, References and Pointers Data types Variables/Data Operations Qt Data Types
file compil build instal librari	Project and compiler settings Compiling, linking and building process Manipulation of Projects Building and installing libraries Configuring QT libraries and executables Compiling and deploy Libraries and Configs

A partir dos dados apresentados, pode-se concluir que o número de capítulos que possuem termos em comum com outros capítulos não é alto. No caso do SWT, existem apenas seis pares de capítulos com intersecção não vazia de termos, o que corresponde a

Tabela 5.8: Sobreposições de termos entre capítulos

Cookbook	Capítulo	Capítulo	# de Termos Comuns
SWT	print file **imag** org.eclipse.e4 pdf	**imag** color draw border font	1
SWT	print **file** imag org.eclipse.e4 pdf	**file** jar eclips icon plugin	1
SWT	print **file** imag org.eclipse.e4 pdf	wizard dispos page tab **file**	1
SWT	thread ui progress ui_thread **call**	creat eclips widget add **call**	1
SWT	creat **eclips** widget add call	file jar **eclips** icon plugin	1
SWT	**file** jar eclips icon plugin	wizard dispos page tab **file**	1
LINQ	type **properti** anonym creat anonym_type	tabl join entiti **properti** relationship	1
LINQ	sql databas entiti **tabl** data	group column row date **tabl**	1
LINQ	**sql** databas entiti tabl data	queri **sql** statement select write	1
LINQ	sql databas **entiti** **tabl** data	**tabl** join **entiti** properti relationship	2
LINQ	group column row date **tabl**	**tabl** join entiti properti relationship	1
LINQ	list item collect **element** arrai	xml file **element** node attribut	1
QT	string qstring pointer type **data**	item model **data** row view	1
QT	**window** platform linux mac process	**window** button dialog click close	1
QT	event mous qml **click** kei	window button dialog **click** close	1
QT	creat http **call** document find	signal slot thread connect **call**	1
QT	**creat** http call document find	file directori path **creat** folder	1
QT	**file** directori path creat folder	**file** compil build instal librari	1

9% de todos as possíveis pares de capítulos (66 pares, i.e., combinação de 12 tomados 2 a 2). Além disso, para todos esses pares de capítulos do SWT, ocorre a repetição de apenas um termo entre os capítulos. No caso do LINQ, também existem seis pares de capítulos com interseção não vazia de termos, o que corresponde a 16,67% de todos os possíveis pares de capítulo. Um desses pares possui dois termos em comum e os outros possuem apenas um termo na interseção. Para o QT, também existem seis pares de capítulos com apenas um termo em comum (9% de todos as possíveis pares de capítulos).

De fato, parece ocorrer sobreposição de temas entre alguns dos pares de capítulos citados acima. Por exemplo, os capítulos "sql databas entiti tabl data" e "group column row date tabl" do LINQ ambos tratam de tabelas/bancos de dados.

RQ2: Em que extensão existe a sobreposição de temas entre os capítulos dos *crowd cookbooks*?
A sobreposição de temas entre capítulos foi analisada através da existência de termos repetidos entre os títulos dos capítulos de cada *cookbook*. O percentual de capítulos com presença de termos repetidos não foi alta (9%, 16,67% e 9% para os *cookbooks* SWT-*Cookbook*, LINQ-*Cookbook* e QT-*Cookbook*, respectivamente). Além disso, em praticamente todos os casos, a repetição é de apenas um termo. **Assim, a resposta à pergunta de pesquisa RQ2 é: o número de capítulos que possuem termos em comum com outros capítulos não é alto**. Como melhoria no sentido de diminuir a sobreposição de temas, destaca-se a necessidade de inclusão de novos termos no conjunto de *Stop Words*.

Percebe-se que alguns dos termos presentes na interseção dos capítulos correspondem a palavras genéricas no contexto de programação (e.g., "call") e que por isso poderiam ser incluídas no conjunto de termos *Stop-Words*, pois assim elas seriam removidas na fase de pré-processamento conduzida antes da aplicação do LDA.

Um outro ponto a se notar é o fato da existência de interseção de termos entre dois capítulos nem sempre indicar a sobreposição de um tema entre eles, devido à polissemia de algumas palavras. Por exemplo, no caso do QT, os capítulos "window platform linux mac process" e "window button dialog click close" possuem o termo "window" em comum. Porém, essas palavras parecem estar sendo usadas com sentido completamente diferentes: sistema operacional Windows e janelas de interface gráfica.

5.5 Respostas ao Critério *Relac*

A Tabela 5.9 mostra o número de receitas que cada um dos participantes avaliou em cada um dos cinco valores (2, 1, 0, -1, -2) possíveis para o critério *Relac*, desconsiderando as receitas *controlled* de cada questionário. Esta tabela também mostra quantidade de capítulos avaliados com nota positiva (2 ou 1), visto que isso é um informação importante, pois corresponde aos casos em que o participante considerou a receita relacionada a pelo menos dois dos cinco termos presentes no título do capítulo ao qual ela pertence. Dos 12 avaliadores considerados na Tabela 5.9, 11 avaliaram pelo menos 26 (i.e., 63,41%) com nota 1 ou 2. Assim como para o critério *AutoCont*, o sujeito 14 também se destacou em relação ao critério *Relac*, com um percentual (35,58%) visivelmente bem inferior ao dos outros avaliadores. O número médio de receitas avaliadas com nota 1 ou 2, dentre os 12 participantes da tabela, é de 31 (75,61%). Considerando a totalidade de avaliações feitas pelos 12 sujeitos (i.e., 492 avaliações, pois cada um dos 12 avaliadores considerados para o critério *Relac*, avaliou um total de 41 receitas), foram 241 (i.e., 48,98%) com nota 2, 131 (i.e., 26,63%) com nota 1, 43 (i.e., 8,74%) com nota 0, 32 (i.e., 6,50%) com nota -1 e 45 (i.e., 9,15%) com nota -2. Assim, pode-concluir que as receitas, em geral, atendem à propriedade de relacionamento com capítulo, pois um percentual significativo das receitas foi considerado totalmente ou parcialmente relacionado com o título do capítulo onde se encontra. Na Figura 5.2 é mostrada informação semelhante à da tabela. Percebe-se que as barras nas cores verde escuro e claro, que representam as notas 2 e 1 respectivamente, são muito maiores do que as outras barras, o que indica que as avaliações foram predominantemente positivas.

5.5.1 Análise dos Comentários

Em relação às observações feitas pelos sujeitos durante avaliação do critério *Relac*, destaca-se o fato de que alguns avaliadores disseram que existem receitas que parecem estar no capítulo errado, i.e., que seria mais adequado se elas estivessem em outro capítulo do *cookbook*. Para saber em qual capítulo uma receita deve ser incluída, conforme explicado na Seção 3.4, estamos utilizando a informação de aderência das *threads* aos tópicos minerados pelo LDA. Uma receita só pode ser incluída em um capítulo se a aderência da

Tabela 5.9: Número de capítulos avaliados por valor de *Relac*

Sujeito	2	1	2 ou 1	0	-1	-2
1	17 (41,46%)	9 (21,95%)	26 (63,41%)	12 (29,27%)	1 (2,44%)	2 (4,88%)
4	23 (56,10%)	11 (26,83%)	34 (82,93%)	5 (12,20%)	2 (4,88%)	0 (0%)
5	29 (70,73%)	3 (7,31%)	32 (78,05%)	1 (2,44%)	1 (2,44%)	7 (17,07%)
6	28 (68,29%)	7 (17,07%)	35 (85,37%)	4 (9,76%)	2 (4,88%)	0 (0%)
7	25 (60,96%)	12 (29,27%)	37 (90,24%)	0 (0%)	2 (4,88%)	2 (4,88%)
8	14 (34,15%)	17 (41,46%)	31 (75,61%)	0 (0%)	5 (12,20%)	5 (12,20%)
10	16 (39,02%)	10 (24,39%)	26 (63,41%)	1 (2,44%)	9 (21,95%)	5 (12,20%)
11	14 (34,15%)	18 (43,90%)	32 (78,05%)	0 (0%)	4 (9,76%)	5 (12,20%)
13	11 (26,83%)	27 (65,85%)	38 (92,68%)	0 (%)	2 (4,88%)	1 (2,44%)
14	6 (14,63%)	9 (21,95%)	15 (35,58%)	16 (39,02%)	2 (4,88%)	8 (19,51%)
15	29 (70,73%)	3 (7,31%)	32 (78,05%)	1 (2,44%)	1 (2,44%)	7 (17,07%)
16	29 (39,02%)	5 (12,20%)	34 (82,93%)	3 (7,31%)	1 (2,44%)	3 (7,31%)

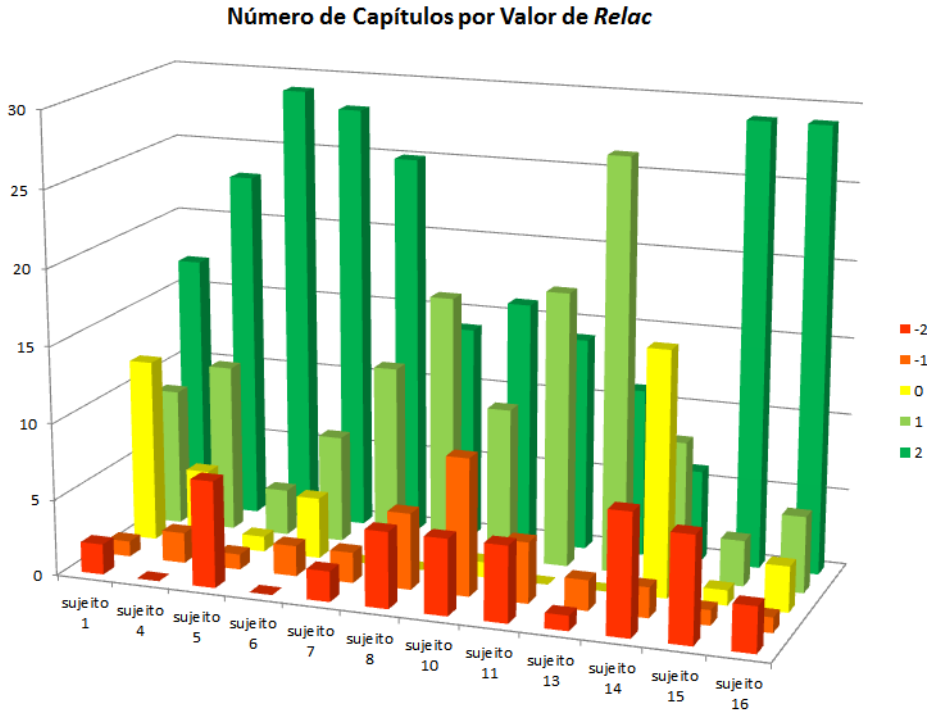


Figura 5.2: Distribuição das notas dos participantes em relação ao critério *Relac*

thread daquela receita ao tópico for no mínimo 51%. Porém, com o uso dessa abordagem ainda assim existem casos de receitas que não são muito aderentes aos temas de seus capítulos. Uma solução para diminuir a ocorrência destes casos seria aumentar o valor do *threshold Ta* que atualmente possui valor de 51%, o que por sua vez iria gerar *cookbooks* com menos receitas, pois mais receitas seriam descartadas por não possuírem aderência suficientes aos tópicos. Outra possível abordagem é empregar métricas de similaridade textual entre os capítulos e os documentos e incluir apenas receitas que possuem similaridade a partir de um certo *threshold*. Essa segunda abordagem foi inclusive utilizada em um trabalho relacionado [27], em que foi utilizado, para propósito semelhante, a medida

de similaridade do cosseno.

RQ3: Em que extensão as receitas dos *crowd cookbooks* estão relacionadas com o tema do capítulo onde se encontram?
A maior parte das receitas (em média 75,61%) foi considerada pelo menos parcialmente relacionado com o capítulo onde se encontra. **Assim, a resposta à pergunta de pesquisa RQ3 é: as receitas, em geral, atendem à propriedade de relacionamento com capítulo**. A partir dos comentários feitos pelos participantes, percebe-se que a principal crítica feita em relação a este critério consiste na existência de receitas mal localizadas, i.e., receitas que estão em determinados capítulos, mas estão mais relacionadas com o tema de outro capítulo. Para melhorar esse ponto foram propostos o uso de métricas de similaridade textual entre receitas e capítulos e alterações no valor do *threshold Ta*, que define a aderência mínima que as receitas devem possuir aos capítulos.

5.6 Respostas ao Critério *Adeq*

A Tabela 5.4 mostra o número de receitas que cada um dos participantes avaliou em cada um dos cinco valores (2, 1, 0, -1, -2) possíveis para o critério *Adeq*, desconsiderando as receitas *controlled* de cada questionário. Esta tabela também mostra quantidade de capítulos avaliados com nota positiva (2 ou 1), visto que isso é um informação importante, pois corresponde aos casos em que o participante considerou a receita, pelo menos parcialmente adequada para ser parte do *cookbook* onde foi incluída. Na Figura 5.3 é mostrada informação semelhante a da tabela. Percebe-se que as barras nas cores verde escuro e claro, que representam as notas 2 e 1 respectivamente, são muito maiores do que as outras barras, o que indica que as avaliações foram predominantemente positivas.

Tabela 5.10: Número de capítulos avaliados por valor de *Adeq*

Sujeito	2	1	2 ou 1	0	-1	-2
4	13 (31,70%)	23 (56,10%)	36 (87,80%)	3 (7,32%)	1 (2,44%)	1 (2,44%)
5	19 (46,34%)	13 (31,71%)	32 (78,04%)	0 (0%)	2 (4,88%)	7 (17,07%)
6	39 (95,12%)	0 (0%)	39 (95,12%)	0 (0%)	2 (4,88%)	0 (0%)
8	27 (65,85%)	11 (26,83%)	38 (92,68%)	0 (0%)	2 (4,88%)	1 (2,44%)
9	6 (16,63%)	18 (43,90%)	24 (58,54%)	9 (21,95%)	4 (9,76%)	4 (9,76%)
10	18 (43,90%)	14 (34,14%)	32 (78,04%)	1 (2,44%)	3 (7,32%)	5 (12,19%)
11	31 (75,61%)	7 (17,07%)	38 (92,68%)	0 (0%)	2 (4,88%)	1 (2,44%)
13	16 (39,02%)	18 (43,90%)	34 (82,93%)	4 (9,76%)	3 (7,32%)	0 (0%)
14	11 (26,83%)	13 (31,71%)	24 (58,54%)	16 (39,02%)	1 (2,44%)	0 (0%)
15	39 (95,12%)	1 (2,44%)	40 (97,56%)	0 (0%)	0 (0%)	1 (2,44%)
16	32 (78,05%)	7 (17,07%)	39 (95,12%)	1 (2,44%)	1 (2,44%)	0 (0%)

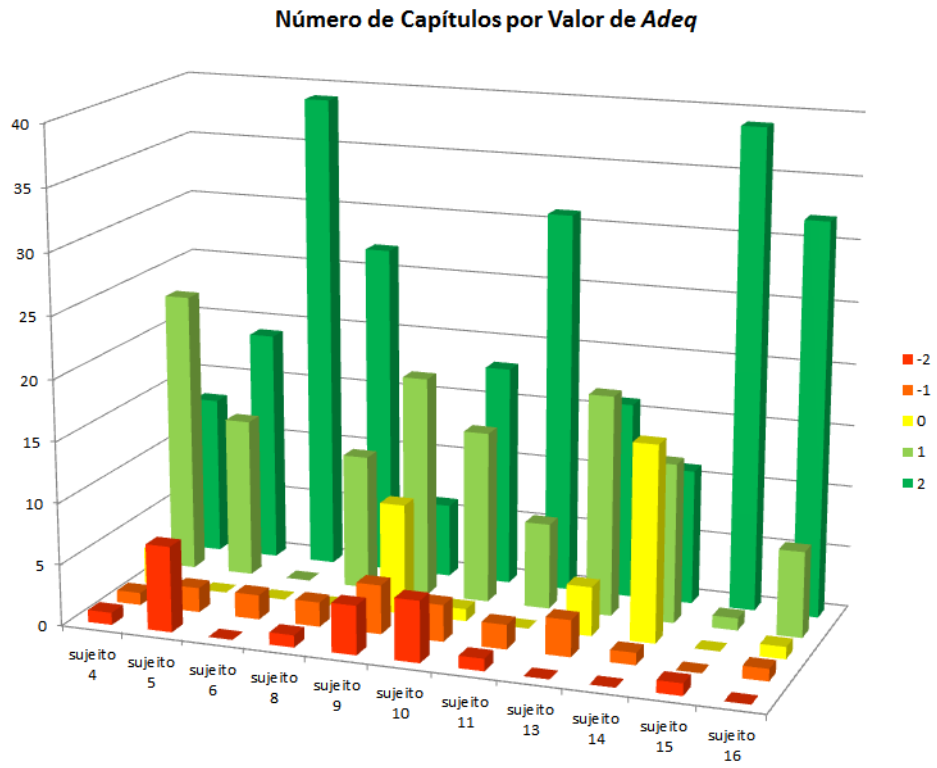


Figura 5.3: Distribuição das notas dos participantes em relação ao critério *Adeq*

Dos 11 avaliadores, todos eles consideraram pelo menos 58,54% (i.e., 24, pois cada participante avaliou 41 receitas) das receitas avaliadas como totalmente adequada ou parcialmente adequada para fazer parte do *cookbook* onde foi incluída. O número médio de receitas avaliados com nota 1 ou 2 é de 34,18, o que corresponde a 83,37% das receitas avaliadas por cada sujeito. Considerando a totalidade de avaliações feitas pelos participantes (i.e., 451 avaliações, pois cada um dos 11 avaliadores considerados para o critério *Adeq*, avaliou um total de 41 receitas), foram 251 (55,65%) com nota 2, 125 (27,71%) com nota 1, 34 (7,54%) com nota 0, 21 (4,66%) com nota -1 e 20 (4,43%) com nota -2. Assim, pode-se dizer que a propriedade de adequabilidade é satisfeita, pois uma porção razoável das receitas foi considerada parcialmente ou totalmente adequada para constituírem um *cookbook*. Pode-se dizer que a estratégia baseada em regras, utilizada para identificar se uma perguntas é um *How-To-Do-It*, possui resultados satisfatórios (lembre-se que na definição do critério *Adeq*, foi explicitado que os pares de pergunta/resposta um *cookbook* devem ser *How-To-Do-It* para serem considerados adequados como receitas).

5.6.1 Análise dos Comentários

Foi feita uma leitura dos comentários feitos pelos avaliadores em relação ao critério *Adeq*. Os pontos citados são mostrados na sequência. A partir dos comentários feitos, percebe-se que os participantes levantaram diversos pontos que levam uma receita a ser adequada ou não para fazer parte de um *cookbook*. Ou seja, apesar de na definição sobre

o critério *Relac* ter sido mencionada apenas a necessidade de a receita ser um *How-To-Do-It*, os avaliadores ampliaram essa noção de adequabilidade. Os principais comentários são colocadas a seguir:

- A necessidade de maiores explicações sobre os trechos de código presentes nas receitas para que as receitas possam ser melhor compreendidas por pessoas novas à API. Por exemplo, um avaliador disse que uma receita " ... *poderia explicar melhor as classes e métodos usados, por exemplo a Image. Mas, para um usuário experiente, que já implementou algo ou parte da aplicação usando a API, acredito que a informação foi suficiente.*". Uma possível abordagem para produzir receitas com uma quantidade maior de explicações/discussões é definir um *threshold* para o tamanho mínimo que a resposta de um par deve ter para o mesmo poder ser considerado uma receita (e.g., "uma resposta deve ter no mínimo N frases em linguagem natural para poder ser incluída num *cookbook*"). Além disso, existe a possibilidade de aplicar o trabalho de Subramanian et al. [60] para identificar e ligar as classes/métodos usados nos trechos de código à documentação oficial da API (e.g., Javadocs). De certa forma, isso consiste em identificar quais são os ingredientes (e.g., classes, métodos) da receita e fornecer explicações adicionais sobre os mesmos;
- A existência de receitas que possuem apenas código-fonte na resposta, sem a presença de explicações em linguagem natural, foi criticada. As explicações em linguagem natural tem importância para criar um contexto para o exemplo de código o que pode facilitar o reuso do exemplo por outros desenvolvedores. No algoritmo de construção de *cookbooks*, exige-se que as respostas tenham código-fonte para serem incluídas no *cookbook*. Talvez fosse importante criar mais uma restrição que imponha a necessidade da presença de texto em linguagem natural;
- A presença de pares de pergunta/resposta cuja pergunta não é um *How-To-Do-It* também foi criticada. Exemplos dessas perguntas incluem perguntas que pedem explicação sobre trechos de código e perguntas sobre aspectos teóricos (conceituais) da computação. Esses casos correspondem a falsos positivos da abordagem baseada em regras que identifica perguntas *How-To-Do-It*. Uma abordagem para resolver este problema é o desenvolvimento de classificadores mais elaborados que tenham maior acurácia na identificação deste tipo de pergunta;
- A presença de *links* referenciando sites externos, com explicações adicionais ou complementares sobre as informações da receita, foi citada como algo benéfico à adequabilidade das receitas;
- A presença de respostas consideradas "grandes demais" foi criticada. No algoritmo de construção de *cookbooks*, pares com perguntas consideradas muito grandes não são incluídos nos *cookbook*. Talvez fosse adequado também evitar a inclusão de respostas muito longas;

- A presença de pares que apresentam pergunta ou resposta muito específicas ao domínio de aplicação do perguntador, foi vista como algo negativo, pois essas receitas não seriam reusadas com tanta frequência, em comparação com receitas mais genéricas;
- O fato de existirem receitas em que os exemplos de código não ficaram claros para o usuário (i.e., trechos de código que eles não entenderam), foi visto como um ponto negativo.

RQ4: Em que extensão as receitas dos *crowd cookbooks* são adequadas para fazerem parte de *cookbooks*?
A maior parte das receitas (em média 83,37%) foi considerada pelo menos parcialmente adequada pelos avaliadores. **Assim, a resposta à pergunta de pesquisa RQ4 é: a propriedade de adequabilidade é satisfeita para a maior parte das receitas**. A partir dos comentários feitos pelos participantes, percebe-se que existem vários fatores que tornam uma receita adequada para ser parte de um *cookbook*: a presença de explicações sobre os elementos da API usados nos trechos de código; a necessidade da pergunta da receita ser um *How-To-Do-It*; a presença de referências a fontes externas para informações complementares; a presença de explicações em linguagem natural; a necessidade das respostas não serem muito grandes; a necessidade das receitas serem genéricas; a necessidade dos trechos de código-fonte estarem claros.

5.7 Respostas ao Critério *AutoCont*

A Tabela 5.11 mostra o número de receitas que cada um dos participantes avaliou em cada um dos cinco valores (2, 1, 0, -1, -2) possíveis para o critério *AutoCont*, desconsiderando as receitas *controlled* de cada questionário. Esta tabela também mostra quantidade de receitas avaliadas com nota positiva (2 ou 1), visto que isso é um informação importante, pois corresponde aos casos em que o participante considerou a receita, pelo menos parcialmente auto-contida. Na Figura 5.4 é mostrada informação semelhante à da tabela. Percebe-se que as barras nas cores verde escuro e claro, que representam as notas 2 e 1 respectivamente, são muito maiores do que as outras barras, o que indica que as avaliações foram predominantemente positivas.

Dos 13 avaliadores considerados na Tabela 5.11, 12 avaliaram pelo menos 37 (i.e., 90,24%) das 41 receitas com nota 1 ou 2. Apenas o sujeito 14 se destacou com um percentual (58,54%) bem inferior aos demais. O número médio de capítulos avaliados com nota 1 ou 2, dentre os 13 participantes apresentados na tabela, é de 36 (87,80%). Considerando a totalidade de avaliações feitas pelos 13 sujeitos (i.e., 533 avaliações, pois cada um dos 13 avaliadores considerados para o critério *AutoCont*, avaliou um total de

Tabela 5.11: Número de capítulos avaliados por valor de *AutoCont*

Sujeito	2	1	2 ou 1	0	-1	-2
3	19 (60,98%)	15 (35,58%)	34 (82,93%)	0 (0%)	7 (17,07%)	0 (0%)
4	33 (80,49%)	1 (2,44%)	34 (82,93%)	0 (0%)	6 (14,63%)	1 (2,44%)
5	17 (41,46%)	17 (41,46%)	34 (82,93%)	4 (9,75%)	3 (7,32%)	0 (0%)
6	35 (85,37%)	6 (14,63%)	41 (100%)	0 (0%)	0 (0%)	0 (0%)
7	35 (85,37%)	5 (12,20%)	40 (97,56%)	0 (0%)	1 (2,44%)	0 (0%)
8	34 (82,93%)	3 (7,32%)	37 (90,24%)	0 (0%)	3 (7,32%)	1 (2,44%)
10	30 (73,17%)	8 (19,51%)	38 (92,68%)	0 (0%)	1 (2,44%)	2 (4,88%)
11	36 (87,80%)	4 (9,76%)	40 (97,56%)	1 (2,44%)	0 (0%)	0 (0%)
12	25 (46,34%)	12 (31,71%)	37 (78,04%)	2 (0%)	2 (4,88%)	0 (17,07%)
13	19 (60,98%)	15 (35,58%)	34 (82,93%)	4 (9,76%)	3 (7,31%)	0 (0%)
14	15 (35,58%)	9 (21,95%)	24 (58,54%)	16 (39,02%)	1 (2,44%)	0 (0%)
15	36 (87,80%)	2 (4,88%)	38 (92,68%)	0 (0%)	3 (7,31%)	0 (0%)
16	34 (82,93%)	3 (7,31%)	37 (90,24%)	1 (2,44%)	2 (4,88%)	1 (2,44%)

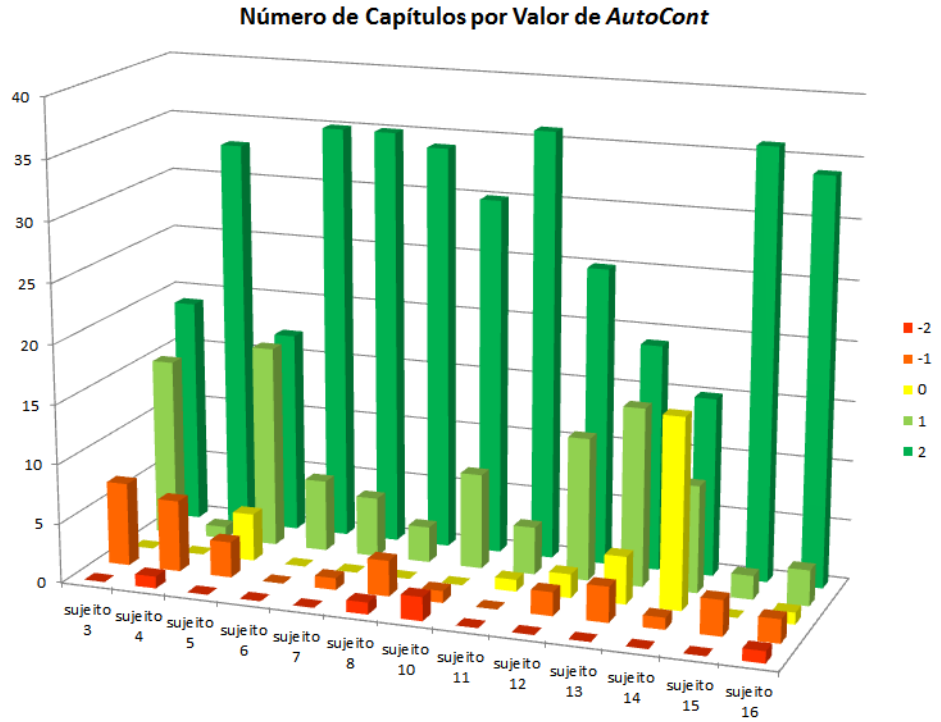


Figura 5.4: Distribuição das notas dos participantes em relação ao critério *AutoCont*

41 receitas), foram 368 (i.e., 69,04%) com nota 2, 100 (i.e., 18,76%) com nota 1, 28 (i.e., 5,25%) com nota 0, 32 (i.e., 6%) com nota -1 e 5 (i.e., 0,94%) com nota -2. Assim, pode-se concluir que a propriedade de auto-contenção, em geral, é satisfeita, pois um grande percentual das receitas foi considerado totalmente ou parcialmente auto-contido. Uma hipótese para explicar os altos valores atingidos para este critério é que são selecionados para serem incluídos nos *cookbooks* apenas pares de pergunta/resposta bem avaliados pela *crowd* (i.e., pares que estão no topo do ranqueamento pelo *score*). Talvez a avaliação feita pela *crowd* em relação ao conteúdo do SO já considere a auto-contenção das soluções

apresentadas, significando que *posts* com alto *score*, tendem a ser auto-contidos.

5.7.1 Análise dos Comentários

Dentre os comentários feitos em relação ao critério *AutoCont*, destacou-se a presença de respostas muito vagas ou resumidas, o que foi considerado um ponto negativo, pois para esse tipo de receita, é necessário buscar informações adicionais para entender completamente a solução apresentada. De certa forma, esse problema coincide com um dos pontos citados em relação ao critério *Adeq*: a necessidade de ter explicações em linguagem natural nas receitas, com o intuito de tornar a solução apresentada mais clara. Uma possível solução para evitar a inclusão de respostas muito resumidas é a criação de um *threshold* que indica o tamanho (e.g., em número de caracteres ou sentenças) mínimo que uma receita deve ter para estar apta a ser incluída num *cookbook*, pois receitas com tamanho pequeno costumam ser muito resumidas e apresentam falta de explicações.

RQ5: Em que extensão as receitas dos *crowd cookbooks* possuem informações auto-contidas?
A maior parte das receitas (em média 87,80%) foi considerada pelo menos parcialmente auto-contida pelos avaliadores. **Assim, a resposta à pergunta de pesquisa RQ5 é: a propriedade de auto-contenção, em geral, é satisfeita**. A partir dos comentários feitos pelos participantes, percebe-se que a presença de receitas muito resumidas prejudica o entendimento das informações nelas contidas, o que faz com que sejam necessárias fontes adicionais de informações para o completo entendimento. Melhorias no sentido de evitar receitas muito resumidas incluem a necessidade da existência de explicações em linguagem natural nas receitas e evitar a inclusão de receita muito pequenas nos *cookbooks*.

5.8 Concordância entre Avaliadores

Com o objetivo de mensurar a concordância entre os vários avaliadores, foi utilizada a métrica *Interclass Correlation Coefficient* (ICC) [33]. Essa medida estatística descreve o quão fortemente as notas dadas por um conjunto de avaliadores para um conjunto fixo de observações se assemelham. Os valores para esta métrica variam no intervalo [0,1]. Quanto mais próximo de 1, maior é a concordância entre as avaliações dos usuários. Foi escolhida esta métrica ao invés do *Weighted Kappa* [14], que é também utilizado para mensurar a concordância para dados ordenados categóricos (que é o caso do estudo aqui desenvolvido, pois os critérios recebem as notas -2, -1, 0, 1 ou 2), porque o *Weighted Kappa* permite o cálculo da concordância apenas entre dois avaliadores, e no caso do estudo desenvolvido neste capítulo, o número de avaliadores é superior a dois.

Para calcular o ICC foi usada a ferramenta IBM SPSS Statistics 2.0[1]. Escolheu-se a opção *two-way-mixed*, porque o objetivo do cálculo não é fazer inferência sobre todos o universo de possíveis avaliadores, e sim apenas sobre os que participaram do estudo. Os valores reportados são os valores de *Single Measure* (o SPSS também calcula uma *Average Measure*), calculados pelo SPSS, pois esta é a medida geralmente utilizada para descobrir se a avaliação de um *rater* é semelhante à dos outros. No cálculo do ICC, para cada critério, foram considerados apenas os sujeitos que avaliaram negativamente os itens *controlled*.

O ICC foi calculado inicialmente entre todos os participantes (considerando apenas as partes comuns aos três tipos de questionários), o que é mostrado na Tabela 5.12. Em seguida, o ICC foi calculado dentre os participantes membros de um mesmo grupo (considerando receitas e capítulos *controlled* e não-*controlled*), o que é mostrado nas Tabelas 5.13, 5.14 e 5.15. O cálculo do ICC foi feito por critério, utilizando um intervalo de confiança de 95%.

Tabela 5.12: Cálculo do ICC dentre participantes dos 3 grupos

Critério	Sujeitos Considerados	ICC (*Single Measure*)	*Lower Bound*	*Upper Bound*
Smt	4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15 e 16	0,331	0,165	0,614
Adeq	4, 5, 6, 8, 9, 10, 11, 13, 14, 15 e 16	0,121	0,054	0,233
AutoCont	3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15 e 16	0,186	0,105	0,315
Relac	1, 4, 5, 6, 7, 8, 10, 11, 13, 14, 15 e 16	0,409	0,287	0,564

Os valores de ICC calculados revelam que a concordância entre os avaliadores não é alta (em geral, a partir do valor 0,60 considera-se que houve uma boa concordância). Assim, pode-se concluir que a concordância entre os avaliadores é baixa ou moderada. Isso confirma que a avaliação dos critérios propostos no estudo é algo extremamente subjetivo. Acredita-se que as diferenças entre os valores de ICC entre os vários critérios são explicadas pelos diferentes graus de subjetividade de cada critério e pelo maior ou menor grau de especificação que foi feito para cada valor da escala *likert* de cada critério. Por exemplo, o critério *Adeq* é o que apresentou o menor valor nas quatro tabelas. A noção de adequabilidade de receitas a *cookbooks* parece ser algo bastante subjetivo. Além disso, na especificação dos cinco valores da escala *likert* deste critério foram usados termos como "totalmente adequada" e "parcialmente inadequada", porém, devido à subjetividade da noção de adequabilidade, não foi definido o significado preciso de cada um destes valores. Já outros critérios, como *Relac* e *Smt*, em geral, apresentaram concordância superior aos outros critérios. Uma possível explicação para esse fato se deve ao fato de as descrições do significado de cada valor da escala *likert* destes critérios terem sido mais precisas. Por

[1]http://http://www-01.ibm.com/software/analytics/spss/products/statistics/

exemplo, no caso do critério *Relac*, as interpretações de cada valor foram definidas em função do número de termos do capítulo que estão relacionados com a receita. O critério *AutoCont* apresentou o segundo menor valor em duas das quatro tabelas. Acredita-se que a atribuição de valores a este critério depende do *background* de cada avaliador em relação à programação em geral, pois isso pode influenciá-lo no grau de entendimento em relação às informações contidas nas receitas.

Tabela 5.13: Cálculo do ICC dentre participantes - Grupo-SWT-Controlled

Critério	Sujeitos Considerados	ICC (*Single Measure*)	*Lower Bound*	*Upper Bound*
Smt	4, 5, 8, 10 e 16	0,550	0,310	0,795
Adeq	4, 5, 8, 10 e 16	0,270	0,146	0,424
AutoCont	4, 5, 8, 10 e 16	0,490	0,355	0,632
Relac	4, 5, 8, 10 e 16	0,423	0,286	0,574

Tabela 5.14: Cálculo do ICC dentre participantes - Grupo-LINQ-Controlled

Critério	Sujeitos Considerados	ICC (*Single Measure*)	*Lower Bound*	*Upper Bound*
Smt	9, 11, 14 e 15	0,309	0,071	0,627
Adeq	9, 11, 14 e 15	0,289	0,130	0,467
AutoCont	3, 11, 12, 14 e 15	0,336	0,201	0,494
Relac	11, 14 e 15	0,443	0,263	0,615

Tabela 5.15: Cálculo do ICC dentre participantes - Grupo-QT-Controlled

Critério	Sujeitos Considerados	ICC (*Single Measure*)	*Lower Bound*	*Upper Bound*
Smt	6, 7 e 13	0,531	0,188	0,805
Adeq	6 e 13	0,267	-0,005	0,509
AutoCont	6, 7 e 13	0,324	0,137	0,515
Relac	1, 6, 7 e 13	0,514	0,366	0,661

RQ7': Em que extensão as opiniões dos sujeitos humanos em relação às propriedades "semântica do capítulo", "relacionamento com tema do capítulo", "adequabilidade para ser parte de *cookbook*" e "auto-contenção" se assemelham?

A concordância entre os avaliadores em relação à propriedades "semântica do capítulo", "relacionamento com tema do capítulo", "adequabilidade para ser parte de *cookbook*" e "auto-contenção" foi mensurada através da métrica ICC. A **resposta à pergunta de pesquisa RQ7' é: os resultados revelam que a concordância é baixa ou moderada**, o que é explicado, em parte, pela subjetividade dos critérios definidos.

5.9 Análise Conjunta dos Critérios *Relac*, *AutoCont* e *Adeq*

É desejável que uma receita tenha bons valores em relação a mais de um critério ao mesmo tempo. Por exemplo, não adianta uma receita estar relacionada com os termos de seu capítulo, mas não ser adequada para estar no *cookbook*. Assim, para os avaliadores em que foram considerados os três critérios *Relac*, *AutoCont* e *Adeq* (i.e., para aqueles que possuem as colunas *Relac*, *AutoCont* e *Adeq* das Tabelas 5.1, 5.2 e 5.3 pintadas de vermelho), foi calculada o número de receitas avaliadas que possuem todos esses três critérios com valor 1 ou 2. O critério *Smt* não foi considerado neste caso porque ele se aplica a capítulos e não receitas. Os resultados são mostrados na Tabela 5.16. Novamente nota-se que o número de receitas avaliadas pelo sujeito 14 com valor 1 ou 2 para os três critérios é bem menor do que para o restante dos usuários. O número médio de receitas avaliadas com nota 1 ou 2 para os três critérios é de 26,8 receitas, o que corresponde a 65,37% do total de receitas avaliadas por cada sujeito. Assim, pode-se concluir que o percentual de receitas que satisfazem simultaneamente aos critérios "'relacionamento com capítulo", "adequabilidade para ser parte de *cookbook*" e "auto-contenção" é razoável, considerando que a abordagem para construção dos *cookbooks* exige apenas uma interação com usuário (o *input* do valor do *threshold maximumPositionAllowed*), sendo totalmente automatizada no restante do processo.

RQ8': Qual a quantidade de receitas que atendem simultaneamente às propriedades "relacionamento com tema do capítulo", "adequabilidade para ser parte de *cookbook*" e "auto-contenção"?
Resposta à pergunta de pesquisa **RQ8': O percentual de receitas avaliadas com nota 1 ou 2 para os três critérios é razoável (média de 65,37% entre os participantes)**.

5.10 Opiniões Gerais dos Avaliadores sobre os *Cookbooks*

As notas globais, na escala de 0 (pior) a 10 (melhor), dadas pelos avaliadores aos *cookbooks*, são mostradas na Tabela 5.17. As células foram coloridas com diferentes tonalidades, sendo o verde mais escuro correspondente à nota máxima (10). O número de opiniões relativas aos três *cookbooks* é diferente porque o tamanhos dos três grupos de avaliadores (Grupo-SWT-Controlled, Grupo-LINQ-Controlled, Grupo-QT-Controlled) não é o mesmo. Além disso, conforme explicado, a opinião do sujeito 2 não está incluída nas tabelas. Algumas células foram pintadas de preto devido a essa diferença do número de

opiniões entre os avaliadores dos três grupos.

Tabela 5.16: Número de receitas com *Relac*, *AutoCont* e *Adeq* $\geqslant 1$

Sujeito	Número de Receitas com *Relac*, *AutoCont* e *Adeq* $\geqslant 1$
4	25 (60,98%)
5	25 (60,98%)
6	35 (85,37%)
8	28 (68,29%)
10	21 (51,22%)
11	29 (70,73%)
13	31 (75,61%)
14	15 (36,59%)
15	30 (73,17%)
16	29 (70,73%)

Tabela 5.17: Notas globais dadas aos *cookbooks*

Sujeito	Nota SWT-*Cookbook*	Sujeito	Nota LINQ-*Cookbook*	Sujeito	Nota QT-*Cookbook*
1	8	1	5	10	4
3	9	4	8	8	8
6	8	5	8	5	10
7	9	6	7	4	8
9	7	7	9	12	8
11	7	8	5	9	7
12	8	10	8	3	8
13	8	13	7	11	7
14	8	16	9	15	9
15	9			14	7
				16	9

As notas médias para os três *cookbooks* são 8,1 (SWT), 7,3 (LINQ) e 7,73 (QT).

A seguir são mostrados os pontos negativos e positivos levantados pelos avaliadores sobre os *cookbooks*, bem como alguns comentários, principalmente em relação aos pontos negativos, no sentido do que poderia ser feito pra minimizá-los.

Os principais **pontos positivos** citados foram:

- *O cookbook é útil para solução de problemas rotineiros*;
- *Presença de perguntas genéricas que podem ser reusadas por outras pessoas que tenham problemas semelhantes*;
- *Presença de códigos-fonte (exemplos).* Isso confirma o senso comum de que a presença de código-fontes é algo considerado importante por desenvolvedores para aprender sobre como resolver tarefas de programação;
- *Algumas receitas mostram o resultado da execução do código (isso permite que o usuário veja rapidamente se o código atende às suas necessidades)*;

- *Proporciona uma visão "legal" sobre a API*;
- *Boa organização/estruturação dos cookbooks*;
- *O agrupamento das receitas, em geral, é satisfatório no que diz respeito ao tema do capítulo*;
- *Presença de capítulos bem definidos*;
- *A organização/estruturação do cookbook em capítulos facilita a localização de informações importantes*;
- *Grande quantidade de informações contidas.* Em outras palavras, os *cookbooks* mostram a solução de uma grande variedade de problemas de programação;
- *Presença de exemplos em que não é preciso acessar sites externos para entendê-los.* Assim, de fato, a auto-contenção das informações contidas nas respostas parece ser algo considerada importante;
- *Fácil de entender para quem tem pouco conhecimento (perguntas simples, respostas de fácil entendimento)*;
- *Presença de respostas que possuem fontes externas que podem ser usadas para buscar mais informações sobre o assunto da receita*;
- *É possível entender a maior parte das informações contidas nos cookbooks*;
- *As informações das receitas se não resolvem o problemas, fornecem um direcionamento para a solução (os códigos-fonte presentes ajudam neste direcionamento)*;
- *Presença de perguntas muito bem estruturadas e de repostas que solucionam o problema, mesmo quando os seus trechos de código não são reproduzíveis*;
- *Presença de vários exemplos tanto de dúvidas iniciais, como também de dúvidas mais avançadas*;
- *Presença de receitas que podem ser testadas sem modificações o que facilita o entendimento do que foi escrito*;
- *Presença de questões muito bem estruturadas.*

Os principais **pontos negativos** citados foram:

- *Presença de perguntas muito específicas, o que dificultaria o reuso das mesmas por outras pessoas.* Para tentar melhorar quanto a esse aspecto, teríamos que estudar maneiras de diferenciar problemas genéricos de problemas muito específicos, de maneira a incluir nos *cookbooks* apenas os genéricos;
- *Algumas receitas poderiam estar em outros capítulos.* As possíveis maneiras de melhorar quanto a este ponto forma discutidas na Seção 5.5;

- *Presença de capítulos mal-definidos (pouco ou nenhum sentido).* Apesar de existirem capítulo mal-definidos, devido aos valores razoáveis obtidos para o critério *Smt*, conforme discutido na Seção 5.3, acredita-que esse não é um problema tão importante;
- *Presença de palavras "cortadas'" (i.e., stems) nos títulos dos capítulos.* Uma possível solução para este problema foi discutida na Seção 5.3.;
- *Presença de capítulos muito grandes (com muitas receitas).* Conforme discutido na Seção 3.6, o número de receitas por capítulos nos *cookbooks* é bastante variável, possivelmente pelo fato de que existem temas mais ou menos populares dentro de uma mesma API. Um estudo de Jiau e Yang [30] confirmou que algumas assuntos de uma API são mais discutidos que outros dentro do SO. Assim, faz sentido termos capítulos grandes para assuntos populares e capítulos pequenos para temas pouco recorrentes. Porém, existe a possibilidade de serem desenvolvidas estratégias para balancear o número de receitas nos capítulos de um *cookbook*;
- *Nem sempre as soluções apresentadas são as melhores, o que torna o cookbook não confiável.* Esse problema é minimizado pelo fato de que são selecionadas apenas receitas que tem alto *score* para serem incluídas nos *cookbooks*. E o *score* é um indicativo de qualidade das informações presentes no SO, pois é um reflexo direto das avaliações feitas pela comunidade do SO;
- *Muitas vezes o perguntador não sabe elaborar muito bem o seu problema (às vezes ele nem fala inglês muito bem).* Para melhorar esse ponto, poderíamos aplicar algumas métricas que indicam a legibilidade de textos. A intuição por trás dessas métricas, é que bom artigos devem ser bem escritos, passíveis de entendimento, e livre de complexidade desnecessária [17]. Assim, somente receitas que fossem consideradas com alto valor de legibilidade seriam incluídas nos *cookbooks*. Exemplos dessa métricas são o *Flesch reading ease* [21] e o *Gunning Fog Index* [24];
- *Apesar de ser possível imaginar o tema que um capítulo abordará, às vezes é difícil saber com certeza que a receita para um problema específico estará nesse capítulo.* *Cookbook* é um tipo de documentação orientado à navegação (exploração). Caso um desenvolvedor tenha um problema específico em mãos e necessita de ajuda para resolvê-lo, a maneira mais eficiente de obter essas informações é recorrer a um mecanismo de busca como Google, o motor de busca do SO ou a sistemas de recomendação desenvolvidos para este propósito (e.g., [19], [51]);
- *Presença de trechos de código bastante minimalistas, que podem ser difíceis de reproduzir para alguém que não conhece a API.* No algoritmo de construção é exigido que as respostas das receitas tenham código-fonte, mas não é feita nenhuma exigência em relação a este código, o que acaba gerando receitas como as da Figura 5.5, com código muito resumidos (nesse caso, o trecho de código consiste de apenas uma

chamada a método, e não mostra, por exemplo, como criar a variável "yourTableViewer"). Uma possível abordagem para resolver esse problema é criar uma restrição que impeça a inclusão de receitas cujo código-fonte possui menos de N LOC (*Lines of Code*). O valor de N seria um *threshold* a ser definido;

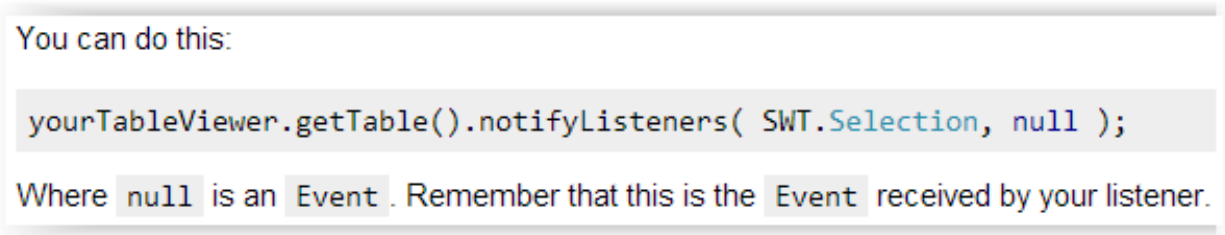


Figura 5.5: Receita com presença de resposta com código minimalista

- *O nome dos capítulos do cookbook não possuem, na sua maioria, informações úteis sobre os itens (receitas) que estão dentro dele.* Esse comentário foi a posição individual do sujeito 14. Conforme mostrado na Seção 5.5, este avaliador se diferenciou por dar notas bastante inferiores aos outros participantes para o critério *Relac*. Como as notas gerais de *Relac* tiveram valores razoáveis, acredita-se que, em geral, as receitas de um capítulo tem a ver com o tema do capítulo, ou seja, que os nomes dos capítulos dão um direcionamento sobre o tipo de receitas que podem ser encontradas dentro dele;
- *Presença de receitas incompletas e que não apontam para outras fontes de conhecimento.* Como mostrado na Seção 5.7, os valores para o critério *AutoCont* foram razoáveis, o que indica que a maior parte das receitas pode ser compreendida sem a necessidade de recorrer a outros recursos. Porém, ainda existe espaço para melhoria. Uma abordagem para evitar receitas com respostas incompletas, é exigir um tamanho mínimo tanto para o código-fonte quanto para o conteúdo em linguagem natural das receitas. Isso tenderia a produzir receitas com respostas maiores e possivelmente mais completas. A presença de referências a fontes externas é vista como algo benéfico pelos usuários de um *cookbook*. As receitas que possuem esse tipo de referência (e.g., *links* para outros *sites*), poderiam ter prioridade para serem incluídas no *cookbook*. Como o algoritmo de construção de *cookbooks* descarta as receitas com *links* quebrados, temos a garantia que somente referências válidas seriam incluídas nos *cookbooks*;
- *Presença de receitas que não estão no formato How-To-Do-It.* Possíveis melhorias para este aspecto já foram discutidas na Seção 5.6;
- *Presença de receitas que não tem correlação com os temas do capítulo.* Possíveis melhoras para este aspecto foram discutidas na Seção 5.5;
- *Muito técnico, difícil de entender para quem não tem muito conhecimento do assunto.* Esse comentário foi feito pontualmente por um avaliador a respeito do *co-*

okbook sobre QT e não foi recorrente entre os demais avaliadores. Porém, uma oportunidade de melhoria que se tira a partir deste comentário é a possibilidade de construção de *cookbooks* para diferentes níveis de experiência (e.g., iniciante, intermediário, avançado). Assim, para construir um *cookbook* para iniciantes, bastaria considerar apenas as perguntas consideradas de iniciantes. Teria que ser elaborada alguma estratégia para classificar uma pergunta em iniciante, intermediária ou avançada, mas acredita-se que isso seja possível a partir de uma análise de padrões textuais (e.g., muitas perguntas de iniciantes contêm frases como: "I'm new to this API", or "Newbie question");

- *A nomeação do capítulo pode ser um pouco confusa, levando o usuário a confundir a questão do tema.* Uma possível solução para melhorar este aspecto é apresentar o título do capítulo como um conjunto de palavras reais ao invés de *stems*. Uma outra possibilidade é recorrer a sujeitos humanos para nomear o título do capítulo (assim como os avaliadores fizeram). Assim, os títulos originais formados por *stems* dariam lugar a uma frase em linguagem natural (e.g., "Working with Colors and Images" ao invés de "imag color draw border font").

Além dos pontos positivos e negativos citados, alguns avaliadores fizeram algumas sugestões de melhoria aos *cookbooks*:

- *Poderia explicar as classes/métodos usados nas receitas, pois usuários menos experientes teriam que recorrer a outras fontes para obter essas informações.* Conforme explicado na Seção 5.6 existe a possibilidade de aplicar o abordagem proposta por Subramanian et al. [60] para identificar e ligar as classes/métodos usados nos trechos de código à documentação oficial da API (e.g., Javadocs). De certa forma, isso consiste em identificar quais são os ingredientes (e.g., classes, métodos) da receita e fornecer explicações adicionais sobre os mesmos. Também poderiam ser evitadas receitas com respostas pequenas demais, já que esse tipo de receita geralmente carece de explicações;
- *Para cada receita, poderia mostrar outras receitas próximas (parecidas) a ela.* A ideia deste avaliador é que apesar de cada capítulo agrupar receitas sob um mesmo assunto geral, seria interessante saber para uma receita quais outras receitas tratam de problemas parecidos. É realmente uma melhoria interessante e simples de ser feita. Poderia se utilizar a medida de similaridade do cosseno para descobrir quais são as receitas mais parecidas com uma em particular. Funcionalidade semelhante já existe no próprio *site* do SO. Quando o usuário está visualizando uma *thread* são indicadas outras *threads* semelhantes (exemplo mostrado na Figura 5.6);
- *Poderia ter uma sequência lógica entre capítulos.* Sequência lógica entre capítulos não é uma característica de *cookbooks*. Porém, esta característica está presente

em outros tipos de documentação, como os tutoriais. Umas das oportunidades de pesquisa derivadas do trabalho apresentado nesta dissertação é utilizar o *crowd knowledge* para construir outros tipos de documentação para APIs como os tutoriais.

RQ9: Quais são os pontos positivos, pontos negativos e melhorias que podem ser feitas nos *crowd cookbooks*?
As notas médias para os três *cookbooks* sobre SWT, LINQ e QT foram 8,1, 7,3 e 7,73 respectivamente. Os avaliadores citaram vários pontos positivos, pontos negativos e também deram sugestões de melhorias, o que nos permite responder à pergunta de pesquisa **RQ9**. Dentre os **pontos positivos** destacam-se: presença de capítulos bem definidos; organização que facilita a localização de informações importantes; presença de respostas que referenciam fontes adicionais de informação; presença de problemas genéricos; existência de código-fonte (exemplos) nas receitas. Dentre os **pontos negativos** destacam-se: presença de perguntas muito específicas; presença de receitas mal localizadas; presença de capítulos com tema mal-definido; presença de *stems* nos títulos dos capítulos; presença de capítulos grandes demais; presença de receitas muito resumidas/incompletas e com código-fonte muito pequeno/incompleto. Dentre as **melhorias** sugeridas destacam-se: indicar para uma receita, outras receitas semelhantes e a inclusão de mais informações sobre os elementos da API usados na receita.

5.11 Opiniões dos Avaliadores sobre Uso dos *Cookbooks* para Aprendizagem

As opiniões completas dos avaliadores sobre o papel dos *crowd cookbooks* no aprendizado das APIs são mostradas no Apêndice D.1. Nas Tabelas 5.18, 5.19 e 5.20 apresentamos uma sumarização das opiniões dadas pelos participantes em relação aos *cookbooks* **SWT-*Cookbook***, **LINQ-*Cookbook*** e **QT-*Cookbook*** respectivamente. Lembrando que os membros do **Grupo-SWT-Controlled** só deram opinião a respeito dos *cookbooks* **LINQ-*Cookbook*** e **QT-*Cookbook***, os membros do **Grupo-LINQ-Controlled** só deram opinião a respeito dos *cookbooks* sobre **SWT-*Cookbook*** e **QT-*Cookbook*** e os membros do **Grupo-QT-Controlled** só deram opinião a respeito dos *cookbooks* sobre **SWT-*Cookbook*** e **LINQ-*Cookbook***. Além disso, a opinião do sujeito 2 foi desconsiderada.

No caso da API SWT, seis avaliadores (sujeitos 3, 11, 12, 13, 14 e 15) disseram que usariam o **SWT-*Cookbook*** para aprender sobre essa API e quatro avaliadores (1, 6, 7 e 9) disseram que não o usariam para este propósito. Percebe-se que apesar de alguns terem respondido que usariam o *cookbook* para aprender e outros que não usariam (devido

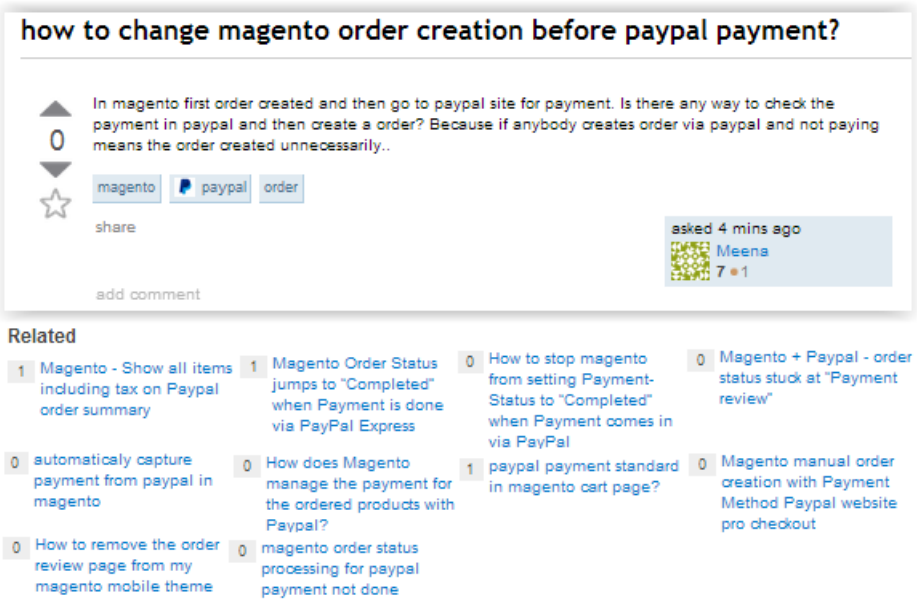


Figura 5.6: *Thread* no SO e outras *threads* relacionadas.

a diferentes interpretações da palavra "aprender"), seis dos dez avaliadores (sujeitos 6, 7, 9, 11, 13 e 14) disseram que utilizariam o *cookbook* apenas para buscar informações sobre problemas específicos. Assim, a percepção que se tem é que o *cookbook* não seria utilizado com uma documentação introdutória, para aprender sobre a API como um todo, do início ao fim. A falta de sequência cronológica entre capítulos e receitas dos *cookbooks* e de um "hello-world" foi citado como algo que dificulta o uso dos *cookbooks* para aprender sobre a API. O fato de o *cookbook* ter diversas informações básicas sobre a API e a presença de códigos-fonte que podem ser executados, foram citados como justificativas para utilizar o *cookbook* para o aprendizado sobre SWT.

Tabela 5.18: Opiniões acerca do uso do *cookbook* **SWT-*Cookbook*** para aprendizado

Sujeitos	Opinião
3	*O cookbook poderia ser utilizado através da execução dos trechos de código das receitas.*
6, 7, 9, 11, 13 e 14	*O cookbook poderia ser utilizado como fonte de consulta para um problema específico, i.e., conforme a necessidade atual do programador.* A percepção destes avaliadores é que o *cookbook* pode ser utilizado para aprender sobre um problema específico e não para aprender sobre a API como um todo.
15	*O cookbook poderia ser utilizado para aprender sobre a API porque ele possui diversas informações básicas sobre a mesma.*
1	*A falta de uma sequência lógica entre os capítulos e de um "hello-word" dificulta o uso dele por pessoas que não conhecem a API.* De fato, *cookbook* é um tipo de documentação que não possui essa característica, ao contrário dos tutoriais.

No caso da API LINQ, cinco avaliadores (sujeitos 4, 5, 8, 10 e 13) disseram que usariam o **LINQ-*Cookbook*** para aprender sobre essa API. Três avaliadores (sujeitos 6, 7 e 16) disseram que não usariam o *cookbook* para aprender sobre LINQ, e sim apenas para tirar dúvidas. O usuário 16 também citou, assim como o sujeito 1 no caso do SWT, que a falta de uma sequência lógica entre capítulos dificulta que o *cookbook* seja usada por um iniciante para aprender sobre a API.

Tabela 5.19: Opiniões acerca do uso do *cookbook* **LINQ-*Cookbook*** para aprendizado

Sujeitos	Opinião
4	*O cookbook poderia ser utilizado para aprender sobre o LINQ porque ele possui dúvidas comuns sobre esta API. A organização do cookbook em capítulos (temas) é intuitivo, o que é uma vantagem em relação ao SO que não possui esta organização.*
5	*O cookbook poderia ser utilizado se algumas receitas trocassem de lugar.* Já foi discutido maneiras de melhorar a aderência das receitas aos capítulos (Seção 5.5);
6, 7, 8, 10, 13 e 16	*O cookbook poderia ser utilizado como fonte de consulta para um problema específico, i.e., conforme a necessidade atual do programador.* A percepção destes avaliadores é que o *cookbook* pode ser utilizado para aprender sobre um problema específico e não para aprender sobre a API como um todo.

No caso do LINQ também nota-se que grande parte dos avaliadores (sujeitos 6, 7, 8, 10, 13 e 16) disseram que o *cookbook* seria utilizado apenas para resolver problemas específicos, apesar de três desses avaliadores terem dito que o *cookbook* poderia ser utilizado no aprendizado (sujeitos 8, 10 e 13) e os outros três terem dito que não (sujeitos 6, 7 e 16). A organização do *cookbook* em temas (capítulos) foi colocado com um fator intuitivo que facilita o aprendizado sobre a API, uma vez que o SO originalmente não possui essa estruturação. A presença de dúvidas comuns (genéricas) também foi citada como um ponto positivo do uso do *cookbook* para aprendizado. Outro ponto de destaque é que um avaliador disse que o *cookbook* teria que melhorar a localização de algumas receitas aos capítulos para ele poder ser usado no aprendizado sobre LINQ.

No caso da API QT, oito avaliadores (sujeitos 3, 4, 5, 8, 11, 12, 14 e 15) disseram que usariam o **QT-*Cookbook*** para aprender sobre essa API. Três avaliadores (sujeitos 9, 10 e 16) disseram que não usariam o *cookbook* do QT para aprender sobre QT, e sim apenas para tirar dúvidas. O sujeito 16 destacou que para o *cookbook* poder ser utilizado para o aprendizado ele deveria organizar as receitas baseando-se no índice de algum livro existente sobre a API, o que está relacionado com a carência de uma ordem cronológica nos *cookbook*, o que não é o caso, por exemplo, de livros didáticos e tutoriais.

No caso do QT também nota-se que uma opinião constante (sujeitos 8, 9, 10, 11 e 16) é que o *cookbook* poderia ser utilizado para obter informações sobre um problema específico e não para aprender sobre a API do início ao fim. Foi citado o fato de que o *cookbook* pode ser utilizado no aprendizado, porém em conjunto com outros recursos, como os tutoriais. Nesse caso, a estruturação do *cookbook* também foi citada como um fator benéfico ao uso do mesmo para aprendizado. A presença de dúvidas básicas (i.e., de iniciantes) foi vista como um facilitador do uso do *cookbook* para aprendizado.

De maneira a geral, a opinião predominante nas avaliações dos três *cookbooks* é que os mesmos podem ser utilizados como fonte de informações para problemas pontuais e não para aprender sobre a API como um todo. Assim, se levarmos em conta a opinião da maioria dos avaliadores, pode-se dizer que os *cookbook* não são adequados para aprender sobre APIs quando o desenvolvedor é totalmente novo em relação a ela.

De certa forma esses resultados corroboram os achados por Robillard [53]. Em um estudo feito com desenvolvedores da Microsoft, ele descobriu que o uso de livros não está entre as principais fontes utilizadas para aprendizado de APIs (as principais estratégias de

aprendizado incluem uso da documentação oficial da API e o uso de exemplos de código). Os *cookbooks* construídos pela abordagem apresentada nesta dissertação podem ser vistos como um tipo de livro, pois eles são inspirados em *cookbooks* "reais" (livros encontrados à venda em livrarias).

Muitos participantes disseram que usariam os *cookbooks* para buscar informações sobre um problema específico. De fato, é possível que o desenvolvedor navegue pelos capítulos e receitas do *cookbook* até localizar a informação que deseja, visto que os *cookbooks* possuem apenas dois níveis de hierarquia (capítulos e receitas), o que facilita a exploração dos mesmos. Ele poderia verificar se o *cookbook* possui um capítulo com tema ligado ao tema de seu problema, e caso exista tal capítulo, vasculhar as receitas contidas nele. Porém, é possível que não exista no *cookbook* uma receita específica para o problema que ele tem em mãos, pois os *cookbooks* são construídos a partir do conteúdo mais bem avaliados pela *crowd* e não têm a preocupação em ser uma documentação que abranja todos os possíveis problemas que podem ser resolvidos usando a API. Além disso, um estudo feito por Parnin et al. [49] mostrou que apesar de o SO ter alto percentual de cobertura em relação a classes de APIs (e.g., 87% das classes do Android são referenciados lá), como esse percentual não é 100%, pode-se dizer que não existe informações no SO sobre a totalidade de cenários que podem ser implementados com a API. E como os *cookbooks* são construídos a partir destes dados, essa característica também está presente neles, ou seja, pode ser que o desenvolvedor não encontre no *cookbook*, informações sobre a tarefa que ele tem em mãos. Adicionalmente, existem maneiras mais eficientes buscar informações sobre um problema específico, como por exemplo, fazer uma pesquisa no Google, no motor de busca disponível no *site* do SO, ou utilizar abordagens de recomendação que utilizam dados do SO [51], [19]. Em outras palavras, as abordagens baseadas em *searching*, onde o usuário pode formular uma *query* que represente a sua necessidade, são mais adequadas para buscar informações pontuais do que a utilização de *cookbooks* que são orientados a *browsing*.

Porém, conforme explicado no Capítulo 3, o processo de *browsing* e *searching* são complementares. O *browsing* pode ser usado em casos em que não é possível a formulação de palavras-chave para compor uma *query*, o que pode ocorrer, por exemplo, quando um usuário é novo em relação a um domínio [47]. Isso vai exatamente de encontro a um comentário feito pelo sujeito 16: "*O conteúdo gerado pelo cookbook pode ajudar o usuário a buscar uma dúvida específica sem que seja necessário um mecanismo de busca, isto é muito importante pois ao iniciar o aprendizado de uma tecnologia muitas vezes existem dúvidas sobre como buscar uma resposta e a estruturação presente no cookbook poderá resolver este problema.*". Assim, como a abordagem de construção de *cookbooks* consiste em fazer uma sumarização das informações disponibilizadas pela *crowd* sobre uma API, uma possível utilização dos *cookbooks* seria no sentido de ajudar desenvolvedores novos a uma API a obterem o conhecimento necessário (i.e., se familiarizarem com os temas e

terminologia da API) para serem capazes de formular *queries* sobre problemas específicos que eles possam vir a ter.

RQ10: Qual o papel que os *crowd cookbooks* podem desempenhar no aprendizado de APIs?
Os dados apresentados nesta seção nos permitem **responder à pergunta de pesquisa RQ10: a opinião mais constante entre os participantes do estudo é que os *crowd cookbooks* produzidos pela estratégia proposta neste dissertação podem ser usados para buscar informações sobre um problema específico, e não para aprender sobre a API como um todo**, principalmente devido à falta de sequência cronológica existente nos *cookbooks*, o que dificulta o uso do mesmo por pessoas novatas à API alvo do *cookbook*. Porém, tiveram opiniões favoráveis ao uso do *cookbook* para aprendizado. Dentre as justificativas para se usar os *cookbooks* no aprendizado de APIs, destacam-se: a intuitiva organização dos mesmos em capítulos; a grande quantidade de informações básicas (i.e., adequadas para iniciantes) e de problemas genéricos nos *cookbooks*; a presença de códigos-fonte que podem ser executados.

5.12 Ameaças à Validade

As ameaças à **validade interna** do estudo desenvolvido nesta dissertação são:

Tabela 5.20: Opiniões acerca do uso do *cookbook* **QT-*Cookbook*** para aprendizado

Sujeitos	Opinião
3	*O cookbook poderia ser utilizado como um fonte para buscar exemplos (códigos-fonte), mas seria necessário recorrer à outras fontes de aprendizado para quem é novo nessa API.*
4	*O cookbook poderia ser utilizado para aprender sobre o LINQ porque ele possui dúvidas comuns sobre esta API. A organização do cookbook em capítulos (temas) é intuitivo, o que é uma vantagem em relação ao SO que não possui esta organização.*
8, 9, 10, 11 e 16	*O cookbook poderia ser utilizado como fonte de consulta para um problema específico, i.e., conforme a necessidade atual do programador.* A percepção destes avaliadores é que o *cookbook* pode ser utilizado para aprender sobre um problema específico e não para aprender sobre a API como um todo.
15	*O cookbook poderia ser utilizado para aprender sobre a API porque ele possui diversas informações básicas sobre a mesma.*
12	*O cookbook poderia ser utilizado para o aprendizado, mas seriam necessárias fontes complementares como tutoriais*
14	*O cookbook poderia ser utilizado para o aprendizado apenas se o usuário tivesse algum conhecimento sobre QT*

- Os participantes dos experimentos são, em sua maioria, conhecidos do autor da dissertação, o que cria um *bias* em relação à avaliação dos mesmos, pois eles poderiam avaliar positivamente os itens dos *cookbooks* apenas na tentativa de beneficiar os resultados da pesquisa. Para mitigar esta ameaça, foram criados *cookbooks controlled* que possuem capítulos e receitas sabidamente ruins. Os participantes que avaliaram com nota neutra ou positiva os itens *controlled* em relação a algum(s) critério(s),

tiveram as avaliações relativas a esse(s) critério(s) descartada(s). Além disso, o fato de os participantes serem voluntários pode significar que eles estavam motivados a participar do estudo;

- Outra ameaça consiste na possibilidade de os participantes terem trocado informações entre si durante o preenchimento do questionário. Porém, acredita-se que essa possibilidade tenha sido minimizada pelo fato que não ter sido publicada a lista de voluntários participantes da pesquisa. Outro fator que mitiga esta ameça é o fato de os questionários poderem ser preenchidos de qualquer computador com acesso à Internet, i.e., não foi necessário agrupar os participantes numa mesma sala, o que reduz a possibilidade de comunicação entre eles;
- O fato de os questionários serem relativamente longos (um total de 213 perguntas) pode ter influenciado no preenchimento correto dos mesmos devido à fatiga dos participantes. Porém, esse problema foi mitigado ao se usar a ferramenta LimeSurvey, pois ela permite que os usuários salvem o progresso do questionário para continuar depois, ou seja, não era necessário que eles completassem todo o questionário de uma única vez. Além disso, acredita-se que o prazo dado aos avaliadores para completar o questionário (2 semanas) foi suficientemente grande, o que contribuiu para que eles fizessem o questionário gradualmente (isso poderia diminuir a fatiga resultante). Apesar disso, nota-se que boa parte dos participantes gastou mais de 2 semanas para o término do questionário;
- Os avalidores foram divididos em três grupos (Grupo-SWT-Controlled, Grupo-LINQ-Controlled e Grupo-QT-Controlled), o que pode ter influenciado nos resultados, pois existem diferenças nos questionários elaborados para cada grupo. Porém, como a divisão dos participantes entre grupos foi aleatória e o perfil deles é muito parecido no que diz respeito ao nível de conhecimento sobre as APIs estudadas, esse problema foi amenizado. Além disso, como o número de participantes foi 16, não foi possível criar os três grupos exatamente com o mesmo tamanho. Porém, os grupos foram criados com tamanhos parecidos (5, 5 e 6), o que minimiza o problema;
- Um outra ameça é o fato de os itens (capítulos e receitas) *controlled* incluídos nos *cookbooks controlled* terem sido escolhidos manualmente pelo autor. Porém, como a maioria dos participantes avaliou esses itens com notas negativas, acredita-se que de fato os itens *controlled* são ruins no que diz respeito aos critérios considerados.

As ameaças à **validade externa** (que afetam a generalização dos resultados obtidos) do estudo desenvolvido nesta dissertação são:

- Os experimentos conduzidos para testar a abordagem propostas no capítulo foram realizados apenas com três APIs, o que compromete a generalização dos resultados

obtidos. Porém, a escolha de APIs ligadas a diferentes linguagens de programação (SWT - Java, LINQ - linguagens .NET e QT - C++) minimiza esta ameaça;

- Um outro fator que afeta a generalização dos resultados obtidos é que apenas 16 pessoas participaram da avaliação dos *cookbooks*, o que não deve corresponder a uma amostra representativa da comunidade de desenvolvimento de software. Prova disso é que o perfil destas pessoas não é muito diversificado no que fiz respeito ao nível de conhecimento sobre as APIs: a grande maioria não possuía nenhum conhecimento sobre as três APIs;
- Outro fator também que afeta a generalização dos resultados é o fato de que os questionários construídos para avaliar os *cookbooks* tratavam de apenas uma amostra dos capítulos e receitas dos *cookbooks*. Para minimizar esta ameaça, o processo de amostragem de receitas selecionou receitas de todos os capítulos dos *cookbooks* de maneira a criar uma amostra mais representativa dos *cookbooks* como um todo. Além disso, a amostragem dos capítulos procurou selecionar capítulos grandes e pequenos, o que contribui para uma maior generalidade.

As ameaças à **validade de construção** são:

- Adivinhação de hipótese (*Hypotesis Guessing*): os participantes do estudo poderiam imaginar que uma das hipóteses do estudo era que os *cookbooks* produzidos pela abordagem proposta poderiam ser usados no aprendizado de APIs e dessa forma responder positivamente quando perguntados sobre isso. Entretanto, a opinião mais recorrente entre os avaliadores é que os *cookbooks* podem ser usados para tirar dúvidas sobre a API e não para aprender sobre a API como um todo. Assim, acredita-se que essa situação não tenha ocorrido para a maioria os avaliadores;
- Expectativa do Experimentador: Como as perguntas discursivas dos questionários foram elaboradas pelo autor da dissertação, pode ser que elas tenham sido influenciadas pelas expectativas do mesmo.

Capítulo 6

Trabalhos Relacionados

Nesta dissertação, apresentamos uma abordagem que faz uso do *crowd knowledge* do SO para construir um tipo de documentação (*cookbooks*) para APIs através da sumarização das informações de APIs disponíveis na *crowd*. Como o SO é um tipo de mídia social, na Seção 6.1 são revisados alguns trabalhos que abordam o impacto e limitações do uso da mídia social em Engenharia de Software. A Seção 6.2 trata de alguns trabalhos que estudaram o processo de aprendizado de APIs por desenvolvedores, o processo de criação e manutenção de documentação, bem como trabalhos que investigaram o potencial do uso do SO para documentação de APIs. Por fim, a Seção 6.3 trata de trabalhos que apresentaram ferramentas/abordagens direcionadas a criação ou melhoria da documentação de APIs. O objetivo dessas seções não é fazer uma revisão sistemática sobre o assunto e por essa razão foram selecionados os trabalhos julgados mais relevantes e/ou relacionados com a abordagem apresentada nesta dissertação.

6.1 Mídia Social na Engenharia de Software

Nesta seção serão apresentados alguns trabalhos que discutem o uso de mídias sociais em Engenharia de Software e o impactos decorrentes deste uso. Com os trabalhos apresentados, pode-se perceber que o uso de mídia social em atividades de Engenharia de Software mudou a forma como os desenvolvedores produzem e compartilham conhecimento sobre desenvolvimento de software. O trabalho apresentado nesta dissertação está profundamente relacionado com os trabalhos apresentados, pois foi apresentada uma abordagem que faz uso das informações disponíveis na mídia social SO.

6.1.1 Storey, Treude, van Deursen & Cheng

Storey et al. [58] defenderam a necessidade de estudos para entender os benefícios, riscos e limitações do uso de mídia social em desenvolvimento de software. No artigo, os autores descrevem a Engenharia de Software como sendo altamente colaborativa, tanto

no nível de times, como de projetos e de comunidade. Isso porque a comunicação de informação é um componente crucial em várias etapas do desenvolvimento de software. Os autores relatam que a geração atual de desenvolvedores faz uso da mídia social, também conhecida como Web 2.0, como forma de suporte ao desenvolvimento colaborativo. No artigo, é relatado que isso representa uma mudança de paradigma, que ocorreu devido a decentralização dos sistemas de computação e devido ao surgimento de uma variedade de ferramentas de mídia social que foram adotadas pela geração atual de desenvolvedores. Os autores descrevem as ferramentas de mídia social como uma "arquitetura de participação", pois elas suportam mecanismos de *crowdsourcing*, bem como um mecanismo de difusão de muitos-para-muitos, isto é, o conteúdo é produzido pela multidão para a multidão. Os autores destacam os principais canais que fazem uso da mídia social em Engenharia de Software, entre eles estão:

- **Wikis**: foram criadas tendo em mente justamente o desenvolvimento colaborativo. Foi um dos primeiros canais de mídia social utilizado em Engenharia de Software e por essa razão seu uso já está bem difundido;
- **Blogs**: são usados frequentemente por desenvolvedores para documentar informação "com-fazer", para discutir o lançamento de novas funcionalidades e para suportar a engenharia de requisitos;
- **Microblogs**: como o Twitter, que são usados constantemente para a troca de informação entre desenvolvedores. Eles fornecem suporte para coordenação e comunicação;
- **Redes Sociais**: permitem a criação de "comunidades virtuais" através de *sites* como o Facebook. Serviços de hospedagem *web* de projetos, como o Github, proveem funcionalidades de redes sociais para mostrar como os desenvolvedores e seus *commits* em várias versões estão relacionados;
- **Crowdsourcing**: através do uso de mídia social, usuários atuam como co-desenvolvedores através do compartilhamento de novos requisitos e *feedback* sobre *bugs*.

Os autores também citam o site de Q&A SO como uma importante forma de troca de informações e gerenciamento de trabalho colaborativo.

Segundo os autores, apesar do uso difundido da Web 2.0 no desenvolvimento de software colaborativo e da grande variedade de mídias sociais usadas, existem poucos estudos empíricos sobre a taxa de adoção e implicações do uso da mídia social na Engenharia de Software.

Os autores delineiam várias perguntas de pesquisa que precisam ser exploradas em trabalhos futuros, para que se possa entender os impactos do uso de mídias sociais na Engenharia de Software. Uma das perguntas elaboradas é: "*How can social media improve individual software development activities?*". O trabalho apresentado nesta dissertação,

de certa forma, está relacionado com esta pergunta, pois foi apresentada uma abordagem que faz uso de mídia social para produzir documentação para APIs e os documentos produzidos foram avaliados por sujeitos humanos. Pode-se assim dizer que o trabalho apresentado nesta dissertação contribui para responder a esta pergunta.

6.1.2 Treude, Figueira Filho, Cleary & Storey

Similarmente ao estudo de Storey et al. [58], Treude et al. [65] discutiram as oportunidades e desafios para os desenvolvedores de software que fazem uso das informações disponibilizadas pela *crowd*, em especial das informações disponíveis no SO. Segundo os autores, o fato de a maior parte (mais de 92%) das perguntas no SO ter tempo mediano de resposta de 11 minutos [40], faz com que o SO seja praticamente um fonte "irresistível" para se obter informações sobre desenvolvimento de software, o que é evidenciado pelos mais de 12 milhões de visitantes e 135 milhões de visualizações de páginas do SO todo mês (esses dados foram obtidos pelos autores em 2012). Os autores acreditam que o SO pode ter um impacto significativo nas práticas de milhões de desenvolvedores pelo mundo todo. O artigo também coloca o SO como o responsável por mudar a maneira como é realizada a busca de informações em sites de Q&A. " *Antes do SO, o principal mecanismo para Q&A eram os fóruns técnicos, onde o conteúdo é disponibilizado através de threads de discussões. O problema dessa abordagem é que as informações úteis ficam misturadas com conteúdo irrelevante. No SO não existe tal problema, pois as respostas podem ser ranqueadas de acordo com a qualidade atribuída pela comunidade por meio do sistema de votação*".

Os autores questionam se é preciso redefinir o conceito de um bom programador: "*Um bom programador é aquele que possui um profundo conhecimento sobre programação e princípios de Engenharia de Software ou alguém que consegue alavancar e sintetizar as informações disponibilizados pela comunidade pode atingir os mesmos resultados?*".

Os autores afirmam que os estudos do que pode ser feito com a mídia social no aprendizado/educação de tecnologias ainda são muito incipientes. O trabalho apresentado nesta dissertação é uma contribuição no sentido de explorar o uso da mídia SO no aprendizado e documentação de APIs.

6.2 Documentação de APIs

Nesta seção serão apresentados os principais trabalhos relacionados a documentação de APIs bem como estudos relacionados ao aprendizado de APIs por parte dos desenvolvedores e a maneira como esses trabalhos se relacionam com esta dissertação.

6.2.1 Parnin & Treude / Parnin, Treude, Grammel & Storey

Parnin e Treude [48] analisaram os resultados de busca no Google para um API particular (JQuery), e descobriram que além das fontes oficiais de documentação, muitos tipos de documentação via mídia social apareceram. Por exemplo, os resultados mostraram que foi obtido pelo menos um *post* de *blog* na primeira página de resultados para 88% dos métodos do JQuery; e para 84% dos métodos, foi encontrada pelo menos uma pergunta no SO. Em um trabalho posterior, Parnin et al. [49] desenvolveram um estudo com o objetivo de compreender o potencial que a *crowd documentation* possui em relação a documentação de APIs. Os autores motivam o possível potencial que a *crowd documentation* possui com a seguinte informação: "*enquanto a documentação oficial para Java disponibiliza um exemplo de código para ilustrar sincronização via o método 'invokeLater()', mas sem texto explanatório e com comentário esparsos, no site de Q&A SO, 286 perguntas podem ser encontradas.*" O estudo desenvolvido no artigo citado investiga a viabilidade de se depender de Q&A sites (em particular, o SO) para obter exemplos e explicações sobre APIs. No estudo foram consideradas três APIs de amplo uso dentre a comunidade de desenvolvimento de software: Android, GWT e Java. O primeiro critério analisado na pesquisa, foi a *Cobertura*, i.e., o percentual de elementos da API (e.g., classes) que são abordados nas *threads* de discussão presentes no SO. Para analisar este critério, os autores desenvolverem um modelo de rastreabilidade entre elementos da API e perguntas/respostas presentes no SO. Os resultados mostraram alta cobertura para as APIs estudadas: por exemplo, 87% das classes da API do Android são referenciadas no SO. Esse alto grau de cobertura sugere que é viável usar o *crowd knowledge* como uma fonte de conhecimento para documentar APIs.

Através do cruzamento de dados de uso de APIs presentes no repositório *Google Code* e os dados de cobertura das classes no SO, os autores encontraram uma forte correlação entre essas duas informações, o que indica que classes que são muito usadas por desenvolvedores, geralmente têm um grande volume de discussão no SO.

Apesar da alta taxa de cobertura encontrada, a velocidade para atingir tais taxas foi linear em relação ao tempo. No estudo também foi analisado a dinâmica de papéis e contribuições no SO. Os resultados apontaram que as dinâmicas de discussões no SO seguem um padrão onde a "multidão" (*crowd*) pergunta e uma quantidade pequena de *experts* respondem a essas perguntas.

Os resultados encontrados por Parnin et al., servem de motivação ao trabalho apresentado nesta dissertação, pois mostrou-se que a comunidade de desenvolvimento (a *crowd*) é capaz de produzir um grande volume de informações relacionadas a APIs, e que essas informações são, em sua maioria, fornecidas por *experts*. Isso encoraja o uso desse conhecimento em outras aplicações, como na estratégia para construção de documentação para APIs que foi apresentada nesta dissertação.

6.2.2 Robillard

Robillard [53] estudou quais são os principais obstáculos ao aprendizado de APIs por parte de profissionais do ramo de desenvolvimento de software. O estudo foi desenvolvido com 80 profissionais de software da Microsoft e foi conduzido através do preenchimento de questionários e da realização de entrevistas. Dentre os participantes do estudo, havia pessoas com diversos níveis de experiência (de iniciante até sênior) e com diferentes cargos (e.g., arquiteto de software, engenheiro de software, programador). Dos 80 participantes, 78% disseram aprender sobre APIs lendo a documentação oficial, 55% através do uso de exemplos de código, 34% através de experimentação com a API, 30% através da leitura de artigos e 29% através de consulta a colegas de trabalho. Além dessas fontes principais de aprendizado, foram citadas outras com baixa frequência, como por exemplo, leitura de livros e depuração do código fonte. Um dos principais resultados do estudo foi que os recursos disponíveis para aprendizado (e.g., documentação) são os principais obstáculos ao aprendizado. Segundo Robillard, este resultado é um indício de que os esforços para melhorar a estrutura de APIs precisam ser complementados por esforços para melhorar os recursos disponíveis para o aprendizado da mesma. O estudo também identificou vários desafios encontrados pelos responsáveis por produzir os recursos usados na documentação de APIs e que podem ser mitigados através da adoção de um série de princípios na construção destes recursos. Os *cookbooks* construídos através da abordagem apresentada possuem alguns dos princípios citados a seguir:

- Incluir bons exemplos de uso da API. A estratégia de construção de *cookbooks* proposta nesta dissertação também se preocupa em incluir bons exemplos de uso da API nas receitas, pois uma das restrições existentes é que as receitas contenham trechos de código. Além disso, como são incluídos nos *cookbooks* pares que possuem alto *score* comparativamente à totalidade de pares da API, os exemplos de código incluídos tendem a ter boa qualidade;
- Considerar a totalidade de elementos presentes na API, ou seja, produzir recursos que sejam completos. Os *cookbooks* produzidos através da abordagem proposta não têm a preocupação de documentar a totalidade das APIs, mesmo porque sabe-se que o *crowd knowledge* do SO não é um corpo de conhecimento completo sobre APIs [49];
- Disponibilizar vários cenários complexos de uso;
- Ser convenientemente organizado/estruturado. Os *cookbooks* produzidos pela abordagem possuem uma estruturação na forma de um conjunto de capítulos;
- Incluir informações relacionadas a aspectos do *design* da API (e.g., explicar a razão de um método existir em uma classe A e não existir em uma classe B).

Outro resultado do estudo, foi que mais de um quarto de todos os participantes identificaram a falta de exemplos de uso da API adequados às suas necessidades, como um obstáculo ao aprendizado da API. No mesmo trabalho, Robillard também investigou como o uso de exemplos, em alguns casos, falha em ajudar os desenvolvedores: "*Exemplos podem ser tornar mais um fator que dificulta do que um recurso ao aprendizado quando existe uma incompatibilidade entre o objetivo do exemplo e o objetivo do usuário. A maior parte das questões relacionadas com exemplos são casos em que os participantes precisavam usar os trechos de código para propósitos mais complexos do que a simples interação com a API.*". Segundo o estudo de Robillard, uma frustração comum em relação ao uso de trechos de código é que, em geral, os mesmos não disponibilizam nenhuma informação sobre como trabalhar com vários elementos da API ao mesmo tempo, ou seja, criar um código mais complexo, que usa, por exemplo, várias classes de uma API. Nos casos em que os trechos de código são insuficiente às necessidades dos desenvolvedores, os mesmos recorrerem a outros recursos como tutoriais ou projetos de software que usam a API em questão. Por fim, Robillard discute a razão de os recursos disponíveis na Internet (dentre os quais está o *crowd knowledge*) ainda não serem a solução definitiva para obtenção de todas as informações necessários ao aprendizado de APIs: 1) a presença de informações de obsoletas e 2) o fato de que exemplos desenvolvidos pelos criadores da API recebem mais credibilidade.

O trabalho apresentado nesta dissertação está relacionado com o estudo desenvolvido por Robillard, pois foi apresentada uma abordagem para criação de um tipo de documentação (*cookbooks*) para APIs e através de um estudo desenvolvido com profissionais/estudantes de pós-graduação, foi avaliado como a documentação criada pode ser usada no auxílio ao aprendizado/compreensão da API. Além disso, as receitas dos *cookbooks* produzidos possuem código-fonte, o que vai de encontro aos resultados apontados pelo estudo de Robillard, em que foi verificada a importância dos exemplos de código-fonte no aprendizado de APIs.

6.2.3 Dagenais & Robillard

Dagenais e Robillard [16] desenvolveram um estudo para entender como a documentação é criada e mantida em projetos de código-livre. Segundo os autores, vários tipos de documentos estão disponíveis para o aprendizado de *frameworks* e bibliotecas, como por exemplo, documentação da API (e.g., Javadocs), tutoriais e manuais de referência, porém pouco se sabe o processo de criação e manutenção deste tipo de documentação. Os autores entrevistarem 22 desenvolvedores que contribuíram com projetos de código-livre e com sua documentação (os chamados *contribuintes*) e também desenvolvedores que frequentemente usaram os projetos e leram sua documentação (os chamados *usuários*). O objetivo era entender como desenvolvedores usam a documentação, e qual tipo de documentação

é mais útil pra eles. Segundo os autores, quando um projeto inicia, os contribuintes têm dois pontos de decisão principais. Primeiro, eles precisam selecionar ferramentas para criar, manter e publicar a documentação. Os autores citam três tipos principais de infraestruturas usadas pelos contribuintes: wikis, *suites* de documentação (e.g., Javadoc) e documentos genéricos como HTML. No trabalho apresentado nesta dissertação, o tipo de documentação produzido (*cookbooks*) não usa nenhuma dessas infraestruturas citadas, pois os mesmos foram publicados em uma aplicação *web* desenvolvida para este propósito, onde é possível visualizar e navegar pelos capítulos/receitas dos *cookbooks*.

Ainda segundo Dagenais e Robillard, o segundo ponto de decisão que os contribuintes encontram inicialmente diz respeito ao tipo de documentação a ser criada. Pela análise das entrevistas conduzidas no estudo, os autores identificaram três tipos de documentação com diferentes focos: 1) a documentação dos "primeiros passos"(*getting started documentation*), que possui como unidade uma tarefa; 2) a documentação de referência, que possui como unidade um elemento de linguagem de programação (e.g., um método); 3) a documentação conceitual que possui como unidade um conceito. Em relação à documentação dos "primeiros passos", os autores explicam que ela possui o objetivo de descrever *como-usar* uma funcionalidade ou conjunto de funcionalidades. Esse tipo de documentação engloba desde pequenos trechos de código, até tutoriais. Os *cookbooks* produzidos usando a abordagem apresentada nesta dissertação também podem ser incluídos nesta categoria, pois as receitas contidas nos mesmos contêm instruções sobre *como-usar* ou *como-fazer* uma tarefa, usando a API em questão. Segundo os resultados apontados por Dagenais e Robillard, este tipo de documentação é geralmente usado como recurso inicial para o aprendizado de projetos/APIs. Os resultados do estudo realizado nesta dissertação para avaliar a utilidade dos *cookbooks* produzidos pela estratégia proposta, mostraram que a maior parte dos avaliadores não considerou o formato dos *cookbooks* adequado como recurso inicial para o aprendizado de APIs. Um dos principais pontos citados pelos avaliadores foi que a falta de uma sequência lógica e progressiva entre os capítulos dos *cookbooks* dificulta o uso deles para aprendizado quando não se tem conhecimento sobre a API. Outro resultado apontado pelo estudo de Dagenais e Robillard foi que outro tipo de documentação que pode ser usado como documentação inicial para o aprendizado, é a documentação de referência. Esse tipo de documentação explica todos os elementos de uma API (e.g, todas suas classes e métodos) e é útil quando uma biblioteca possui em sua maioria, funções atômicas, pois pode ser difícil criar exemplos de código-fonte que possuam chamada a muitas funções.

Nas discussões dos resultados, Dagenais e Robillard dizem acreditar que ferramentas que produzem/incrementam a documentação existente podem ser úteis, mas que não podem ser substituídas por documentação escrita por profissionais humanos. No trabalho apresentado neste dissertação, foi avaliado em que extensão, um tipo de documentação (*cookbooks*) produzido via uma abordagem semi-automatizada pode ser útil ao desenvol-

vedor.

6.3 Ferramentas de Auxílio à Documentação

Nesta seção, serão apresentadas alguns trabalhos que propuseram ferramentas que produzem ou incrementam a documentação de APIs. Estes trabalhos estão relacionados com a abordagem para construção de *cookbooks* apresentada neste dissertação, uma vez que os *cookbooks* produzidos também constituem um tipo de documentação para APIs. Além dos trabalhos detalhados nesta seção, foram encontrados na literatura, diversos outros trabalhos relacionados a ferramentas de auxílio à documentação [9], [56], [10], [15], [59]. Os trabalhos detalhados na seção forma escolhidos com base no relacionamento com a abordagem apresentada nesta dissertação para geração de *cookbooks*. Porém, é possível ter uma visão geral da assunto a partir dos trabalhos revisados. Além disso, ao melhor do meu conhecimento, a proposta apresentada para construção de *cookbooks* é pioneira. Assim, apesar de existir trabalhos relacionados, não existe nenhum trabalho que possua exatamente o mesmo objetivo.

6.3.1 Long, Wang & Cai

Long et al. [39] apresentaram *Altair*, uma ferramenta que automaticamente gera referências cruzadas de funções de APIs. Os autores motivaram o trabalho dizendo que apesar de documentação de APIs, como as páginas do "man" do Unix, e os Javadocs possuírem informações de referência cruzada entre os elementos da API (e.g., ao visualizar a documentação para uma API, em geral existe um "SEE ALSO" para verificar quais outras funções em geral são usadas para resolver uma tarefa relacionada ou parecida), manter esse tipo de documentação é uma tarefa dolorosa e que demanda muito trabalho. Além disso, os autores destacam que apesar de algumas ferramentas de documentação como o *doxygen*, e o compilador C# conseguirem criar documentação com as informações de referência cruzada, a partir de anotações em código-fonte feitas pelo desenvolvedor, em geral, é difícil para os desenvolvedores rastrearam todas os elementos da API relacionados a uma função.

A ferramenta *Altair* realiza uma análise estática para extrair informações estruturais do código-fonte, e a partir desses dados calcula a sobreposição de pares, i.e., as duplas de funções podem estar relacionadas entre si. A entrada para utilização da ferramenta é uma *query* que contém o nome de uma função. A saída é um conjunto de pares que estão relacionados com a função dada como *input*.

Foram realizados experimentos que mostraram que *Altair* superou outras ferramentas de recomendação de APIs, e que ela é capaz de clusterizar as funções de uma API em grupos bem distintos entre si.

Uma possibilidade para incrementar a abordagem de construção de *cookbooks* apresentada nesta dissertação é empregar a estratégia de Long et al. para informar ao usuário do *cookbook* quais são os métodos de APIs relacionados aos que foram usados nos trechos de código-fonte da receita.

6.3.2 Kim, Lee, Hwang & Kim

Kim et al. [31] propuseram uma técnica para automaticamente incrementar a documentação de APIs (e.g., Javadocs) com exemplos de código-fonte.

Segundo dados apresentados no artigo, a maior parte da documentação de APIs não contém exemplos de código-fonte suficientes. Em uma análise feita pelos autores sobre os Javadocs de mais de 27.000 elementos de APIs (e.g., classes), apenas 2% (em torno de 500 elementos) deles incluíam exemplos de código. Segundo os autores, esse fato faz com os desenvolvedores tenham que recorrer a ferramentas de busca de código-fonte, como o *Ohloh*[1] e o *Google Code*[2]. Porém, é destacado que os desenvolvedores possuem dificuldades em encontrar exemplos apropriados ao usar estas ferramentas: muitas vezes o desenvolvedor tem que formular várias *queries* e analisar os resultados das mesmas até encontrar algo que corresponda às suas necessidades, o que acaba por consumir muito de seu tempo. Para resolver esse problema, os autores citam que existem trabalhos direcionados à extração de exemplos de código de repositórios [29, 70, 71]. Segundo os autores, uma desvantagem destas abordagens é que são necessárias ferramentas (e.g., Eclipse) para que se possa acessar os exemplos. Em contraste, os autores afirmam que incrementar a documentação de APIs é vantajoso por duas razões: 1) a documentação de APIs é o primeiro lugar onde os desenvolvedores recorrem para obter informações de uso; 2) os desenvolvedores não necessitam de ferramentas adicionais para acessar esta documentação.

As contribuições do trabalho foram:

- Uma técnica automática para geração de exemplos de código: Eles propuseram uma técnica que extrai exemplos de código de um repositório, extrai características semânticas dos mesmos, clusteriza esses exemplos, e para cada cluster seleciona o exemplo de código-fonte mais representativo;
- A inserção dos exemplos selecionados nos Javadocs das APIs desses elementos;
- Uma avaliação para mensurar a quantidade de exemplos de código-fonte gerados.

Segundo os resultados apresentados no artigo, foi possível incluir exemplos de código para mais 20.000 elementos de APIs (75% de todos os elementos considerados no estudo).

O trabalho apresentado nesta dissertação, apesar de estar relacionado com o artigo de Kim et al., pois ambos propõem técnicas direcionadas à documentação de APIs, possui

[1] https://code.ohloh.net/
[2] https://code.google.com/

várias diferenças. A primeira diferença que se nota é em relação ao tipo de documentação alvo dos dois trabalhos: enquanto Kim et al. propuseram o incremento de Javadocs, o alvo do trabalho apresentado nesta dissertação é um tipo diferente de documento (*cookbooks*). Outra diferença que se nota é que enquanto Kim et al., propuseram uma técnica para incrementar uma documentação já existente (Javadocs), no trabalho apresentado nesta dissertação é proposta uma abordagem para criar uma documentação do zero, i.e., os *cookbooks* produzidos não são resultado da melhoria de uma versão anterior de *cookbooks*. Outra diferença que se nota é que no trabalho aqui apresentado, o conteúdo usado para construir os *cookbooks* é minerado a partir do *crowd knowledge* existente no SO, enquanto que no trabalho de Kim et al., os exemplos são obtidos a partir de repositórios genéricos de código-fonte, como o *OhLoh* (que na época de escrita do artigo era conhecido como *Koders*). A fraqueza desta abordagem é que o trechos de código extraídos não têm nenhum contexto explicando o que eles estão tentando alcançar. Em contraste, os trechos de código criados pela multidão, em geral, não possuem esta desvantagem, pois muitas vezes eles estão acompanhados por texto explicativo e discussão em linguagem natural.

6.3.3 Dekel & Herbsleb

Dekel e Herbsleb [20] desenvolveram um *plugin* para o Eclipse chamado *eMoose*, que destaca no código-fonte métodos que estão sendo usados e que possuem "diretivas" na sua documentação. Os autores definem diretivas como sendo regras ou ressalvas existentes na documentação da API. Por exemplo, uma diretiva pode conter a seguinte informação: o método A só deve ser chamado caso o método B tenha sido chamado, caso contrário ocorrerá um erro em tempo de execução. Essa informação pode estar contida na documentação da API, mas é, em muitos casos, ignorada pelo desenvolvedor, o que pode originar um código problemático.

O objetivo do trabalho descrito no artigo é fazer com que os desenvolvedores possam examinar um fragmento de código-fonte consciente das diretivas associadas àquele trecho. Os autores acreditam que a consciência dessas diretivas, pode tanto evitar erros de invocação, como também auxiliar os desenvolvedores a aprender a API a partir de exemplos de código-fonte.

Os autores determinaram algumas regras para definir o que é uma diretiva, isto é, as informações que devem ser "empurradas a força" na consciência do programador: 1) elas devem conter passos concretos que o desenvolvedor pode seguir de uma maneira direta; 2) elas devem capturar informação inesperada, não frequente e não trivial. As diretivas usadas nos experimentos do artigo foram identificadas pelos autores a partir da leitura do texto explicativo de APIs contido em Javadocs (em geral os Javadocs possuem um texto em linguagem natural que explica sobre seu elemento alvo (e.g., uma classe)). O *plugin* *eMoose* destaca no código-fonte que está sendo desenvolvido, os elementos da API que

possuem diretivas na descrição de seus Javadocs. Ao clicar neste elementos, o *plugin* abre uma janela contendo a descrição da API para aquele elemento. As diretivas existentes são mostradas em destaque nesta janela.

Para validar a proposta apresenta no artigo, foi conduzido um estudo de laboratório comparativo no qual desenvolvedores foram solicitados a corrigir pequenos erros de código. O grupo de desenvolvedores que usou o *eMoose* teve melhores resultados do que o grupo que não o usou.

No trabalho apresentado nesta dissertação, o tipo de documentação alvo é *cookbook*, enquanto a abordagem de Dekel e Herbsleb é focada em Javadocs. Outra diferença, diz respeito ao fato de que *eMoose* ser um *plugin* para um IDE. Os *cookbooks* produzidos nesta dissertação são apresentados por si só, pois eles são inspirados na ideia de livros *coobooks* existentes comercialmente. Existe a possibilidade de incrementar a abordagem apresentada nesta dissertação para a produção de *cookbooks* com a estratégia usada pelo *eMoose*, de maneira a destacar nos trechos de código-fonte das receitas, os elementos que possuem diretivas em sua documentação. Entretanto, um fator que dificulta isso é o fato de que as diretivas terem que ser manualmente identificadas via a leitura da documentação das APIs, o que prejudica a escalabilidade da abordagem.

6.3.4 Henb, Monperrus & Mezini

Henb et al. [27] apresentaram uma técnica semi-automatizada para a construção de FAQs (*Frequently Asked Questions* ou Perguntas Frequentemente Perguntadas) a partir dados provenientes de discussões sobre desenvolvimento de software, como listas de *e-mails* e fóruns. No artigo, os autores destacam a popularidade das FAQs, que são usadas como parte da documentação de vários projetos de software diferentes, como Linux[3], Apache Lucene[4] e Eclipse SWT[5]. A proposta apresentada no artigo é pioneira em construir FAQs, apesar de existirem outros trabalhos relacionados à produção de outros tipo de documentação de software (como os apresentados nas seções 6.3.2 e 6.3.3). A proposta para construção de *cookbooks*, apresentada nesta dissertação, também é pioneira, porém possui vários pontos em comum com o trabalho de Henb et al., que serão destacados adiante.

A proposta de Henb et al. extrai FAQs a partir de canais como listas de *e-mails* e comunidades de Q&A através da identificação de tópicos populares e recorrentes e encontrando pares de perguntas e respostas representativos para estes tópicos. A proposta apresentada nesta dissertação para a construção de *cookbooks* também procede desta forma, porém usa apenas dados provenientes do site de Q&A SO. A proposta para criação dos FAQs é dita semi-automatizada porque os FAQs criados podem ser validades

[3]http://tldp.org/FAQ/Linux-FAQ/index.html
[4]http://wiki.apache.org/lucene-java/LuceneFAQ
[5]http://www.eclipse.org/swt/faq.php

e editados (e.g., reformulação para maior claridade) por profissionais humanos, antes de serem publicados. A estratégia para construção de *cookbooks* proposta nesta dissertação, também é semi-automatizada, pois exige interação com o usuário para obter o valor de um dos *thresholds* usados no algoritmo. Existe a possibilidade de empregar mecanismos de edição e validação similares à técnica para geração de FAQs na estratégia de construção de *cookbooks*. Uma diferença que se percebe entre *cookbooks* e FAQs é que *cookbooks* são voltados para problemas práticos que desenvolvedores podem deparar. Por essa razão, apenas são incluídos nos *cookbooks*, pares de perguntas e respostas que contêm instruções sobre como realizar uma tarefa usando a API (i.e., *How-To-Do-It*). Pela análise dos FAQs produzidos pela estratégia de Henb et al.[6], percebe-se os FAQs incluem vários tipos diferentes de perguntas, ou seja, várias categorias diferentes da categoria *How-To-Do-It*, apesar de a categoria *How-To-Do-It* também estar presente. Outro diferença entre os trabalhos é que as receitas dos *cookbooks* produzidos possuem trechos código-fonte, pois a intenção é que as receitas mostrem como solucionar um problema, usando elementos da API. Na estratégia para geração de FAQs, não preocupa-se em selecionar conteúdo com código-fonte para ser incluído na documentação.

Para validação da proposta, Henb et al. geraram FAQs para 50 grandes projetos de código-livre presentes no repositório *OhLoh* e convidaram os principais desenvolvedores desses projetos para avaliar os FAQs produzidos. Entre os resultados encontrados, destaca-se que 82% das respostas selecionados pelo algoritmo de geração de FAQs foram consideradas corretas pelos especialistas. No trabalho apresentado nesta dissertação foram produzidos *cookbooks* para APIs e não projetos de software, mas também foram conduzidos experimentos com sujeitos humanos para avaliar a documentação produzida.

Existem várias semelhanças entre as duas abordagens, no que diz respeito à metodologia para produzir a documentação. Entre elas, pode-se citar: 1) o emprego de mecanismos semelhantes de pré-processamento dos documentos (antes da aplicação do LDA); 2) o uso do LDA (*Latent Dirichlet Allocation*) para encontrar os tópicos existentes em um conjunto de documentos; 3) a remoção de tópicos com poucos pares de perguntas e respostas associados (apesar de o *threshold* usado nos dois trabalhos ter sido diferente). Além das diferenças mencionadas anteriormente, existem outras diferenças entre as duas abordagens: 1) o trabalho de Henb et al. descarta pares de pergunta e respostas que não são muito relacionados ao seu tópico do LDA, usando para isso a medida de similaridade do cosseno. Na estratégia para produção de *cookbooks*, utiliza-se para este propósito a informação de distribuição probabilística dos documentos em relação aos tópicos (i.e., a aderência dos documentos aos tópicos), que é um dos resultados da aplicação do LDA; 2) na abordagem de Henb et al., ranqueia-se os pares de pergunta/resposta dentro de um tópico, usando a similaridade que cada um possui com os termos do tópico. No trabalho para produção dos *cookbooks*, o ranqueamento das receitas considera o *score* dos pares,

[6] http://faqcluster.com/

i.e., são incluídos apenas os pares mais bem avaliados pela *crowd*.

6.3.5 Subramanian, Inozemtseva & Holmes

Subramanian et al. [60] apresentaram um método para ligar elementos de código-fonte com a documentação da API.

Segundo os autores, o uso de bibliotecas de terceiros pode reduzir significantemente o esforço necessário para desenvolver um novo sistema. Porém, "*entender como usar essas APIs pode ser complicado. O uso da documentação da API pode ser uma maneira valiosa para o entendimento da API, porém geralmente ela é insuficiente por si só. Um dos principais problemas é que a documentação geralmente está desatualizada*". Por essa razão, os desenvolvedores ignoram a documentação e se interessam mais por exemplos de código. Como resultado desta situação desenvolvedores recorrem a recursos *online* como o SO. Ainda conforme apresentado no artigo, raramente existem *links* entre os recursos *online* e a documentação oficial da API: a documentação não contém *links* para os exemplos e os exemplos raramente contém *links* para a documentação. Para resolver este problema, o artigo apresenta uma técnica para identificar referências a tipos, chamadas de métodos, referências a campos em trechos de código-fonte. Os autores demonstraram a generalidade da abordagem ao aplicá-la para linguagens tipadas (no caso Java) e dinâmicas (no caso JavaScript).

Nos experimentos mostrados no artigo, a ferramenta obteve alta precisão (97% dos tipos presentes nos trechos de código foram corretamente identificados) e foi capaz de ligar com sucesso os trechos de código-fonte a várias classes e métodos (no caso do Java) e funções (no caso do Javascript) presentes na documentação destas APIs.

Nesta dissertação, foi apresentada uma abordagem para construção de *cookbooks* para APIs a partir dos dados do SO. As receitas presentes nos *cookbooks* contêm trechos de código-fonte. Uma possibilidade de melhorar a abordagem proposta nesta dissertação é empregar a técnica apresentada por Subramanian et al. para ligar os tipos de dados que são referenciados nos segmentos de código à documentação oficial da API.

6.3.6 Montandon, Borges, Felix & Valente

Montandon et al. [45] desenvolveram o *APIMiner*, uma plataforma que incrementa a documentação de APIs para Java (i.e., os Javadocs) com exemplos concretos de uso, extraídos de um repositório privado de código. A implementação do *APIMiner* é baseada num algoritmo de sumarização que utiliza uma técnica de *static slicing* para extrair exemplos de código-fonte pequenos, mas relevantes.

Os autores construíram uma instância do *APIMiner* para a API Android. No total foram extraídos 73.732 exemplos de código-fonte a partir de 103 projetos *open-source*. Foi feito um estudo usando esta versão para Android, no qual desenvolvedores usaram o

sistema por um período de quatro meses. O sistema foi acessado mais de 20.000 vezes, a partir de 130 países, com um total de mais de 42.000 visualizações.

No trabalho apresentado nesta dissertação, o tipo de documentação alvo é *cookbook*, enquanto a abordagem de Montandon et al. é focada em Javadocs. Além disso, ao contrário do trabalho de Montandon et al., nesta dissertação foi apresentada uma abordagem para construir documentação do zero, e não melhorar a documentação já existente. Outra diferença entre os trabalhos é que enquanto a fonte de dados do *APIMiner* são repositórios de projetos de software, a abordagem apresentada nesta dissertação utiliza o *crowd knowledge* disponível no SO.

Capítulo 7

Conclusões e Trabalhos Futuros

Nesta dissertação foi apresentada uma abordagem semi-automatizada que faz uso das informações disponíveis no *site* de perguntas e respostas SO para a construção de *cookbooks* (um tipo de documentação orientado a receitas) para APIs. As motivações para o desenvolvimento do estudo aqui apresentado são a falta de exemplos e explicações em boa parte das documentação oficial de APIs e a tendência nas duas últimas décadas do uso de mídia social em atividades da Engenharia de Software. O SO é atualmente uma das principais vias pelos quais desenvolvedores compartilham conhecimento sobre desenvolvimento de software, o que gerou um importante corpo de conhecimento que é criado e mantido pela *crowd*, sendo os principais contribuintes do conteúdo postado no SO *experts* em programação de software. Os *cookbooks* produzidos pela estratégia proposta são chamados de *crowd cookbooks* e devem ser usados através de um processo de navegação (*browsing*).

A estratégia de construção de *cookbooks* faz uso da técnica de modelagem de tópicos LDA pra encontrar os principais temas existentes sobre uma certa API no *crowd knowledge* do SO. Cada tópico minerado pelo LDA potencialmente dá origem a um capítulo no *cookbook* sendo construído. Os capítulos são preenchidos com receitas, i.e., pares de perguntas/resposta do SO que contêm um cenário e instruções sobre como resolvê-lo utilizando elementos da API. Apenas um tipo especial de perguntas (as *How-To-Do-It*) são de interesse dos *cookbooks* e por isso foi elaborada uma estratégia baseada em regras com o objetivo de identificar este tipo de pergunta.

Foram definidas propriedades desejáveis de *cookbooks*: os títulos dos capítulos devem ter semântica definida; os títulos dos capítulos devem ter semânticas que não se sobreponham; as receitas devem ser adequadas para serem parte de *cookbook*; as receitas devem estar relacionadas com os termos presentes no título de seu capítulo; e as receitas devem possuir informações auto-contidas. Foi realizado um estudo para verificar se os *cookbooks* produzidos pela estratégia proposta atendem a essas propriedades. O estudo também procurou descobrir qual o papel que *cookbooks* podem desempenhar no aprendizado de APIs.

Para testar a abordagem proposta, foram gerados *cookbooks* para três importantes APIs: SWT, LINQ e QT. Um grupo de sujeitos humanos composto por profissionais do ramo de desenvolvimento de software e estudantes de pós-graduação em Ciência da Computação avaliou os *cookbooks* gerados para as APIs.

Dentre os resultados encontrados, destacam-se que os avaliadores consideraram que um percentual considerável dos capítulos dos *cookbooks* possuem sentido bem definido (média de 59,75% dos capítulos avaliados pelos participantes). Outro resultado interessante encontrado foi que os temas percebidos pelos vários avaliadores para um dado capítulo foram muito parecidos entre si. A maior parte das receitas também foi considerada adequada para estarem nos *cookbooks* onde foram incluídas (média de 83,37% das receitas avaliadas por cada participante). Além disso, os participantes também consideraram que a maior parte das receitas possuem informações auto-contidas (média de 87,80% das receitas avaliadas por cada participante) e estão relacionados com o termos do capítulo onde estão (média de 75,61% das receitas avaliadas por cada participante).

A opinião mais constante entre os avaliadores é que os cookooks não são adequados para serem utilizados para o aprendizado de uma API como um todo, principalmente pelo fato de não existir uma sequência progressiva, i.e., uma ordem cronológica entre as receitas e capítulos. Por outro lado, uma opinião recorrente entre os participantes é que os *cookbooks* podem ser usados para buscar informações sobre um problema específico. Apesar de originalmente os *cookbooks* terem sido projetados para serem usados através de uma estratégia de exploração (navegação) e não de busca, acredita-se que eles podem ser úteis para que um desenvolvedor novato a uma API possa se familiarizar com os termos usados na API e a partir disso ser capaz de utilizar uma estratégia voltada à busca (*search*) para obter informações mais pontuais sobre um problema específico que ele tenha interesse. Trabalhos futuros são necessários para melhor entender como os *crowd cookbooks* podem ser usados na prática por desenvolvedores de software.

O estudo apresentado nesta dissertação de certa forma contribui para entender como o *crowd knowledge* pode ser usado em Engenharia de Software, em especial na documentação de APIs, pois foi apresentada uma abordagem para construção de documentação para APIs. Dagenais e Robillard [16] acreditam que o uso de ferramentas para produzir ou incrementar a documentação de APIs não pode substituir a documentação escrita por seres humanos. De fato, a qualidade de um *cookbook* feito por humanos é claramente superior a dos *crowd cookbooks*. Porém, não é para todo tipo de tecnologia ou API que existem versões comerciais de *cookbooks*. Em contrapartida, o aspecto dinâmico e interativo do SO, favorece a discussão de novas tecnologias e APIs, à medida que elas passam a ser utilizadas pelos desenvolvedores. Assim, os *crowd cookbooks* podem ser especialmente úteis para APIs que carecem de documentação oficial e/ou de *cookbooks* produzidos/editados por humanos.

Devido à crescente importância do SO nas práticas de desenvolvimento de software,

não se pode ignorar o importante corpo de conhecimento disponibilizado nessa mídia social. Deve-se alavancar essa vasta quantidade de informações e o trabalho apresentado nesta dissertação é uma contribuição neste sentido.

7.1 Trabalhos Futuros

O primeiro trabalho futuro que poderia ser derivado a partir da abordagem para construção de *cookbooks*, consiste em modificar a abordagem para atender às críticas feitas pelos avaliadores. Os principais pontos a serem modificados seriam:

- Melhorias nos títulos dos capítulos dos *cookbooks*. Uma crítica constante foi que os títulos contém "palavras cortadas" (*stems*) ao invés de palavras reais. Assim, seria necessária uma modificação no algoritmo para mapear os *stems* utilizados no LDA para palavras originais;
- Melhorias no sentido de garantir que as receitas estão relacionadas com os capítulos onde elas são incluídas. Até o momento, apenas a informação de aderência gerada pelo LDA está sendo usada com esse objetivo. Uma possibilidade é também incluir outras métricas, como a similaridade do cosseno para este objetivo;
- Definir maneiras de identificar receitas com respostas mais completas. Uma crítica existente foi a existência de receitas muito resumidas, tanto no que diz respeito às explicações em linguagem natural, quanto ao trechos de código-fonte contidos, que às vezes são muito pequenos. Assim, seria necessário definir maneiras de identificar receitas que possuem quantidade razoável de explicações, bem como um trecho de código não muito resumido;
- Existem capítulos com tamanhos muito variados nos *cookbooks* devido à diferença de popularidade entre assuntos para uma mesma API, o que acaba por refletir na diferença de informações disponíveis na *crowd* para os vários assuntos de uma mesma API. Como essa característica não foi bem vista por alguns dos avaliadores, existe uma oportunidade de melhoria no sentido de balancear os tamanhos dos capítulos de um *cookbook*;
- Melhorias para identificar as perguntas do tipo *How-To-Do-It*. Na abordagem atual está sendo utilizada uma simples abordagem baseada em regras, que leva em conta os termos presentes no título e corpo das perguntas. Essa abordagem deve ser melhor testada para que possamos calcular métricas que indicam a sua performance de classificação, como por exemplo, *recall*, precisão, especificidade, média harmônica. Isso é importante para verificarmos a quantidades de falsos positivos ou falsos negativos da abordagem, e possivelmente refinar as regras utilizadas na abordagem. Uma outra possibilidade para identificação deste tipo de pergunta é utilizar um

classificador baseado em Regressão Linear recentemente desenvolvido no grupo de pesquisa do LASCAM [1] [11].

Em relação às melhorias propostas por alguns avaliadores, como trabalhos futuros poderiam ser implementadas:

- A identificação dos elementos da API (e.g., classes e métodos) que são referenciados nos trechos de código das receitas e ligação destes elementos a outros fontes de informação, como por exemplo, a documentação oficial daquela API. Com essas informações adicionais, o entendimento das informações das receitas pode ser facilitado, especialmente para usuário novos à API;
- A identificação, para cada receita, de outras receitas que tratam de problemas parecidos. Atualmente, as receitas agrupadas sob um mesmo capítulo tratam de um mesmo tema geral. Porém, dentro de um mesmo capítulos podem existir *clusters* de problemas parecidos entre si. Encontrar esse relacionamento de menor granularidade entre as receitas seria uma forma de incrementar a organização atual dos *cookbook* o que poderia facilitar a exploração das informações para um usuário que está navegando pelo *cookbook*.

Uma extensão do trabalho apresentado nesta dissertação consiste na produção de *cookbooks* para níveis de *expertise* variados. Por exemplo, poderiam ser construído *cookbooks* apenas com receitas mais básicas, mais simples de serem compreendidas por novatos.

Um oportunidade de pesquisa derivado deste trabalho é a possibilidade de utilizar as informações da *crowd*, em especial do SO, para construir outros tipos de documentação para APIs, em especial tutoriais. Pela opinião de alguns avaliadores, percebe-se que a ordem cronológica existente entre os capítulos de um tutorial é algo importante para o aprendizado de APIs, o que é uma característica ausente nos *cookbooks*. A maior dificuldade para a construção de tutoriais parece ser como estabelecer uma ordem cronológica entre os tópicos.

A construção de um *cookbook* para uma API consiste em fazer uma sumarização das informações disponibilizadas pela *crowd* sobre a API, sendo que o interesse nesse caso são apenas em perguntas do tipo *How-To-Do-It*. O conhecimento da *crowd* também pode ser sumarizado para outros propósitos. Um dos tipos de perguntas que são feitas no SO são as *Debug-Corrective*, que são relacionadas a problemas de compilação e/ou execução de um código-fonte. Assim, uma possibilidade é utilizar este tipo de pergunta em uma abordagem de sumarização semelhante à conduzida para construção dos *cookbooks*. O resultado seria uma espécie de catálogo sobre os principais problemas de programação envolvendo a API, e eles também estariam agrupados por temas comuns. Porém, seria necessária a identificação automática deste tipo de pergunta. Outra possibilidade é sumarizar o

[1] http://lascam.facom.ufu.br/

conhecimento da *crowd* para criar FAQs (*Frequently Asked Questions*) sobre APIs. As FAQs abordam diversos tipos diferentes de perguntas e não apenas as *How-To-Do-It* (desde que as perguntas sejam frequentes). Um trabalho relacionado à construção de FAQs já foi proposto [27], porém nos experimentos realizados no estudo foram geradas FAQs apenas para projetos de software e não para APIs. A geração de FAQs para APIs poderia gerar resultados interessantes.

Os *cookbooks* produzidos através da abordagem proposta nesta dissertação foram apresentados aos avaliadores sem nenhum tipo de pós-processamento para melhorar a qualidade dos mesmos. Uma possível abordagem consiste em antes de liberar os *cookbooks* para o público final, apresentá-los a *experts* na API alvo do *cookbooks*. Esses profissionais poderiam fazer uma série de alterações nos *cookbook* para melhorar a qualidade dos mesmos, como por exemplo:

- Trocar o título dos capítulos (conjunto de 5 *stems*) por uma frase em linguagem natural;
- Eliminar capítulos com sentido mal definido;
- Eliminar receitas que não são *How-To-Do-It*;
- Eliminar receitas que estão mal localizadas.

A desvantagem desta abordagem é que ela tornaria todo o processo de construção e lançamento de um *cookbook* muito mais lento e menos automatizado.

Outro trabalho futuro consiste em estudar em um ambiente real de desenvolvimento de software (e.g., uma empresa de TI) como a estratégia de *browsing*, em especial os *cookbooks*, podem ser usados no dia-a-dia de desenvolvedores.

Outro trabalho relacionado diz respeito à avaliação dos *cookbooks* gerados por profissionais que tenham experiência com as APIs consideradas, pois no estudo apresentado nesta dissertação, todos os 16 avaliadores tinham pouca ou nenhuma experiência com as APIs estudas. Com essa avaliação adicional, poderíamos inclusive considerar o critério *Reprod*, que foi desconsiderado no estudo atual. Outra possibilidade de trabalho futuro é identificar outras características desejáveis de *cookbook* e verificar se os *cookbooks* produzidos pela estratégia proposta possuem essas características.

Outro trabalho futuro consiste em elaborar alguma maneira de descobrir o valor do *threshold maximumPositionAllowed* automaticamente, pois a determinação do valor deste *threshold* é o único ponto da abordagem que exige algum tipo de intervenção humana. Isso tornaria a técnica totalmente automatizada.

Outro trabalho futuro consiste em produzir *cookbooks* para versões específicas de APIs. As *threads* sobre uma API no SO podem estar relacionadas a diversas versões daquela API. Como podem existir mudanças significativas entre uma versão e outra (e.g., quando uma nova versão de uma biblioteca é lançada, métodos da versão anterior podem se tornar

obsoletos) é interessante produzir *cookbooks* que contenham receita que usam apenas um versão da API. Essa melhoria vai de encontro a um dos achados do estudo de Robillard [53]: a presença de informações obsoletas é algo que prejudica o aprendizado de APIs. O principal desafio de implementar essa abordagem é como identificar sobre qual versão cada *thread* está tratando.

Na abordagem proposta nesta dissertação, como *proxy* para a qualidade de um *post* está sendo utilizada o valor do *score*. Essa abordagem tem uma limitação, pois *posts* mais antigos tendem a ter maior *score* do que novos, mesmo que estes possuam qualidade superior. Assim, são necessários mecanismos para prever a qualidade de *posts* independente da idade dos mesmos. Já existem trabalhos com esse objetivo [17], [2] e cujas ideias poderiam ser aproveitadas. Além disso, outra métrica que poderia ser utilizada no processo de seleção dos *posts* é o número de visualizações das suas respectivas *threads* no SO, visto que isso é um indicativo da popularidade das mesmas e acredita-se que as dúvidas mais populares e recorrentes são as mais adequadas para serem incluídas em *cookbooks*.

Referências Bibliográficas

[1] M. Alex and D. Ascber. *Python Cookbook*. O'Reilly Media, 2002.

[2] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering Value from Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 850–858, New York, NY, USA, 2012. ACM.

[3] D. Bajic and K. Lyons. Leveraging Social Media to Gather User Feedback for Software Development. In *Proceedings of the 2Nd International Workshop on Web 2.0 for Software Engineering*, Web2SE '11, pages 1–6, New York, NY, USA, 2011. ACM.

[4] A. Barua, S. W. Thomas, and A. E. Hassan. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empirical Software Engineering*, 19:619–654, 2012.

[5] O. Barzilay, C. Treude, and A. Zagalsky. *Facilitating Crowd Sourced Software Engineering via Stack Overflow*, pages 297–316. Springer, New York, 2013.

[6] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[7] G. Bougie, J. Starke, M.-A. Storey, and D. M. German. Towards Understanding Twitter Use in Software Engineering: Preliminary Findings, Ongoing Challenges and Future Questions. In *Proceedings of the 2Nd International Workshop on Web 2.0 for Software Engineering*, Web2SE '11, pages 31–36, New York, NY, USA, 2011. ACM.

[8] F. P. Brooks, Jr. *The Mythical Man-month (Anniversary Ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[9] M. Bruch, M. Mezini, and M. Monperrus. Mining subclassing directives to improve framework reuse. In J. Whitehead and T. Zimmermann, editors, *MSR*, pages 141–150. IEEE, 2010.

[10] R. P. Buse and W. R. Weimer. Automatic Documentation Inference for Exceptions. In *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, ISSTA '08, pages 273–282, New York, NY, USA, 2008. ACM.

[11] E. Campos and M. Maia. Automatic categorization of questions from Q&A sites. In: ACM Symposium on Applied Computing. In *Proc. of the 29th Symposium On Applied Computing*, SAC '14, 2014.

[12] W. Chang. *R Graphics Cookbook*. O'Reilly Media, 2012.

[13] H. Chen, A. L. Houston, R. R. Sewell, and B. R. Schatz. Internet Browsing and Searching: User Evaluations of Category Map and Concept Space Techniques. *J. Am. Soc. Inf. Sci.*, 49(7):582–603, May 1998.

[14] J. Cohen. Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit., 1968.

[15] B. Dagenais and H. Ossher. Automatically Locating Framework Extension Examples. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT '08/FSE-16, pages 203–213, New York, NY, USA, 2008. ACM.

[16] B. Dagenais and M. P. Robillard. Creating and Evolving Developer Documentation: Understanding the Decisions of Open Source Contributors. In *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE '10, pages 127–136, New York, NY, USA, 2010. ACM.

[17] D. H. Dalip, M. A. Gonçalves, M. Cristo, and P. Calado. Exploiting User Feedback to Learn to Rank Answers in Q&A Forums: A Case Study with Stack Overflow. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 543–552, New York, NY, USA, 2013. ACM.

[18] I. Darwin. *Java Cookbook*. O'Reilly Media, 2001.

[19] L. B. L. de Souza, E. C. Campos, and M. d. A. Maia. Ranking Crowd Knowledge to Assist Software Development. In *Proceedings of the 22Nd International Conference on Program Comprehension*, ICPC 2014, pages 72–82, New York, NY, USA, 2014. ACM.

[20] U. Dekel and J. D. Herbsleb. Improving API Documentation Usability with Knowledge Pushing. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 320–330, Washington, DC, USA, 2009. IEEE Computer Society.

[21] R. Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32(3):p221 – 233, June 1948.

[22] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, Nov. 1984.

[23] T. L. Griffiths and M. Steyvers. Finding Scientific Topics. *PNAS*, 101(suppl. 1):5228–5235, 2004.

[24] R. Gunning. *The Technique of Clear Writing*. McGraw-Hill, New York, 1952.

[25] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized Recommendation of Social Software Items Based on Social Relations. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 53–60, New York, NY, USA, 2009. ACM.

[26] T. Hattori. Wikigramming: A Wiki-based Training Environment for Programming. In *Proceedings of the 2Nd International Workshop on Web 2.0 for Software Engineering*, Web2SE '11, pages 7–12, New York, NY, USA, 2011. ACM.

[27] S. Henb, M. Monperrus, and M. Mezini. Semi-automatically Extracting FAQs to Improve Accessibility of Software Development Knowledge. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 793–803, Piscataway, NJ, USA, 2012. IEEE Press.

[28] R. Hoffmann, J. Fogarty, and D. S. Weld. Assieme: Finding and Leveraging Implicit References in a Web Search Interface for Programmers. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 13–22, New York, NY, USA, 2007. ACM.

[29] R. Holmes and G. C. Murphy. Using structural context to recommend source code examples. In *Proceedings of the 27th ICSE*, pages 117–125. ACM, 2005.

[30] H. C. Jiau and F.-P. Yang. Facing Up to the Inequality of Crowdsourced API Documentation. *SIGSOFT Softw. Eng. Notes*, 37(1):1–9, Jan. 2012.

[31] J. Kim, S. Lee, S.-w. Hwang, and S. Kim. Adding Examples into Java Documents. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, pages 540–544, Washington, DC, USA, 2009. IEEE Computer Society.

[32] D. E. Knuth. *The art of computer programming, volume 1 (3rd ed.): fundamental algorithms*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.

[33] G. G. Koch. *Intraclass Correlation Coefficient*, pages 213–217. John Wiley & Sons, New York, 1982.

[34] R. Kuc. *Apache Solr 4 Cookbook*. Packt Publishing, 2013.

[35] A. Kuhn, S. Ducasse, and T. Gírba. Semantic clustering: Identifying topics in source code. *Inf. Softw. Technol.*, 49(3):230–243, Mar. 2007.

[36] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen. A Study of the Difficulties of Novice Programmers. *SIGCSE Bull.*, 37(3):14–18, 2005. ACM, 2005.

[37] D. Lancaster. *TTL cookbook*. Sams, Indianapolis, IN, 1974.

[38] S. Laurent. *Jquery Cookbook*. O'Reilly Media, 2009.

[39] F. Long, X. Wang, and Y. Cai. Api Hyperlinking via Structural Overlap. In *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ESEC/FSE '09, pages 203–212, New York, NY, USA, 2009. ACM.

[40] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann. Design lessons from the fastest Q&A site in the west. In *Proceedings of CHI*, pages 2857–2866, New York, NY, USA, 2011. ACM.

[41] G. Marchionini and B. Shneiderman. Finding facts vs. browsing knowledge in hypertext systems. *IEEE Computer*, pages 70–80, 1988.

[42] A. K. McCallum. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu, 2002.

[43] A. Mockus, R. T. Fielding, and J. Herbsleb. A case study of open source software development: the Apache server. In *Proc. of the 22nd ICSE*, pages 263–272. ACM, 2000.

[44] M. Monperrus, M. Eichberg, E. Tekes, and M. Mezini. What should developers be aware of? An empirical study on the directives of API documentation. *Emp. Softw. Eng.*, 17(6):703–737, 2012. Springer US, 2012.

[45] J. E. Montandon, H. Borges, D. Felix, and M. T. Valente. Documenting APIs with examples: Lessons learned with the APIMiner platform. In *WCRE*, pages 401–408. IEEE, 2013.

[46] S. Nasehi, J. Sillito, F. Maurer, and C. Burns. What Makes a Good Code Example? A Study of Programming Q&A in Stack Overflow. In *Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 25–34, 2012.

[47] C. Olston and E. H. Chi. ScentTrails: Integrating Browsing and Searching on the Web. *ACM Trans. Comput.-Hum. Interact.*, 10(3):177–197, Sept. 2003.

[48] C. Parnin and C. Treude. Measuring API documentation on the web. In *Proc. of the 2nd Web2SE*, pages 25–30. ACM, 2011.

[49] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey. Crowd Documentation: Exploring the Coverage and the Dynamics of API Discussions on Stack Overflow. *Georgia Tech, Tech. Rep.*, 2012. IEEE Comp. Soc., 2012.

[50] A. Polukhin. *Boost C++ application development cookbook*. Packt Publ., 2013.

[51] L. Ponzanelli, A. Bacchelli, and M. Lanza. Leveraging Crowd Knowledge for Software Comprehension and Development. In *CSMR*, pages 57–66, 2013.

[52] M. F. Porter. Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[53] M. P. Robillard. What Makes APIs Hard to Learn? Answers from Developers. *IEEE Softw.*, 26(6):27–34, 2009. IEEE Computer Society Press, 2009.

[54] E. Sadun. *The core ios 6 developers cookbook*. Addison-Wesley, 2013.

[55] J. Sametinger. *Software Engineering with Reusable Components*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[56] G. Sridhara, E. Hill, D. Muppaneni, L. Pollock, and K. Vijay-Shanker. Towards Automatically Generating Summary Comments for Java Methods. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 43–52, New York, NY, USA, 2010. ACM.

[57] D. Stephens, D. Christopher, J. Turkanis, and J. Cogswell. *C++ Cookbook*. O'Reilly Media, 2006.

[58] M.-A. Storey, C. Treude, A. van Deursen, and L.-T. Cheng. The Impact of Social Media on Software Engineering Practices and Tools. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, pages 359–364, New York, NY, USA, 2010. ACM.

[59] J. Stylos, B. A. Myers, and Z. Yang. Jadeite: Improving API documentation using usage information. In *CHI Extended Abstracts*, pages 4429–4434. ACM, 2009.

[60] S. Subramanian, L. Inozemtseva, and R. Holmes. Live API Documentation. In *TO APPEAR Proc. of the 36rd ICSE*, page 11. ACM, 2014.

[61] A. Sureka, A. Goyal, and A. Rastogi. Using Social Network Analysis for Mining Collaboration Data in a Defect Tracking System for Risk and Vulnerability Analysis. In *Proceedings of the 4th India Software Engineering Conference*, ISEC '11, pages 195–204, New York, NY, USA, 2011. ACM.

[62] C.-M. Tan, Y.-F. Wang, and C.-D. Lee. The Use of Bigrams to Enhance Text Categorization. *Inf. Process. Manage.*, 38(4):529–546, July 2002.

[63] S. W. Thomas. Mining software repositories using topic models. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 1138–1139, New York, NY, USA, 2011. ACM.

[64] C. Treude, O. Barzilay, and M.-A. Storey. How Do Programmers Ask and Answer Questions on the Web? (NIER Track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 804–807. ACM, 2011.

[65] C. Treude, F. F. Filho, B. Cleary, and M.-A. D. Storey. Programming in a Socially Networked World: the Evolution of the Social Programmer. The Future of Collaborative Software Development (FutureCSD), 2012.

[66] G. Unmesh. *Selenium Testings Tools Cookbook*. Packt Publishing, 2012.

[67] D. ČubraniĆ, G. C. Murphy, J. Singer, and K. S. Booth. Learning from Project History: A Case Study for Software Development. In *Proceedings of the 2004 ACM Conference on CSCW*, pages 82–91. ACM, 2004.

[68] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1105–1112, New York, NY, USA, 2009. ACM.

[69] D. Wolff. *Open GL 4.0 Shading Language Cookbook*. Packt Publishing, 2011.

[70] T. Xie and J. Pei. MAPO: Mining API usages from open source repositories. In *Proceedings of the 2006 international workshop on Mining Software Repositories*, pages 54–57. ACM, 2006.

[71] H. Zhong, T. Xie, L. Zhang, J. Pei, and H. Mei. MAPO: Mining and Recommending API Usage Patterns. In *Proceedings of the 23rd European Conference on ECOOP 2009 — Object-Oriented Programming*, Genoa, pages 318–343, Berlin, Heidelberg, 2009. Springer-Verlag.

Apêndice A

Lista de Stop Words

A.1 Stop Words Usadas no Pré-Processamento

- **Lista de *stop-words* da língua inglesa**: a, about, above, accordingly, across, after, afterwards, again, against, all, allows, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anywhere, apart, appear, appropriate, are, around, as, aside, associated, at, available, away, awfully, b, back, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, below, beside, besides, best, better, between, beyond, both, brief, but, by, c, came, can, cannot, cant, cause, causes, certain, changes, co, come, consequently, contain, containing, contains, corresponding, could, currently, d, day, described, did, different, do, does, doing, done, down, downwards, during, e, each, eg, eight, either, else, elsewhere, enough, et, etc, even, ever, every, everybody, everyone, everything, everywhere, ex, example, except, f, far, few, fifth, first, five, followed, following, for, former, formerly, forth, four, from, further, furthermore, g, get, gets, given, gives, go, gone, good, got, great, h, had, hardly, has, have, having, he, hence, her, here, hereafter, hereby, herein, hereupon, hers, herself, him, himself, his, hither, how, howbeit, however, i, ie, if, ignored, immediate, in, inasmuch, inc, indeed, indicate, indicated, indicates, inner, insofar, instead, into, inward, is, it, its, itself, j, just, k, keep, kept, know, l, last, latter, latterly, least, less, lest, life, like, little, long, ltd, m, made, make, man, many, may, me, meanwhile, men, might, more, moreover, most, mostly, mr, much, must, my, myself, n, name, namely, near, necessary, neither, never, nevertheless, new, next, nine, no, nobody, none, noone, nor, normally, not, nothing, novel, now, nowhere, o, of, off, often, oh, old, on, once, one, ones, only, onto, or, other, others, otherwise, ought, our, ours, ourselves, out, outside, over, overall, own, p, particular, particularly, people, per, perhaps, placed, please, plus, possible, probably, provides, q, que, quite, r, rather, really, relatively, respectively, right, s, said, same, second, secondly,

see, seem, seemed, seeming, seems, self, selves, sensible, sent, serious, seven, several, shall, she, should, since, six, so, some, somebody, somehow, someone, something, sometime, sometimes, somewhat, somewhere, specified, specify, specifying, state, still, sub, such, sup, t, take, taken, than, that, the, their, theirs, them, themselves, then, thence, there, thereafter, thereby, therefore, therein, thereupon, these, they, third, this, thorough, thoroughly, those, though, three, through, throughout, thru, thus, time, to, together, too, toward, towards, twice, two, u, under, unless, until, unto, up, upon, us, use, used, useful, uses, using, usually, v, value, various, very, via, viz, vs, w, was, way, we, well, went, were, what, whatever, when, whence, whenever, where, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, whoever, whole, whom, whose, why, will, with, within, without, work, world, would, x, y, year, years, yet, you, your, yours, yourself, yourselves, z, zero, i'm, you're, he's, she's, it's, we're, they're, i've, you've, we've, they've, i'd, you'd, he'd, she'd, we'd, they'd, i'll, you'll, he'll, she'll, we'll, they'll, isn't, aren't, wasn't, weren't, hasn't, haven't, hadn't, doesn't, don't, didn't, won't, wouldn't, shan't, shouldn't, can't, couldn't, mustn't, let's, that's, who's, what's, here's, there's, when's, where's, why's, how's;

- **Lista de *stop-words* da específica do domínio do SO**: actual, answer, app, application, approach, code, development, help, implement, need, program, project, question, require, response, soluttion, source, suggestion, sure, system, thank, thing, think, understand, user, want, algorithm, class, execute, function, main, method, object, problem, result, set, return, run, note, look, try, interface, ok, idea.

Apêndice B

Documentos Usados no Treinamento

B.1 Documento de Treinamento

As Figuras B.1, B.2, B.3, B.4, B.5, B.6 e B.7 mostram o documento de treinamento fornecido aos participantes do estudo.

Documento de Treinamento Para Preenchimento de Questionário

1. Objetivo

O objetivo deste documento é instruir os participantes da pesquisa sobre como preencher o questionário alvo da pesquisa. Para tanto será necessário fazer uma série de definições que são dadas nas próximas seções.

2. Definições

2.1 Cookbook: É um tipo de documentação para **APIs** (*Application Program Interface* ou Interface para Programação de Aplicações), como por exemplo, bibliotecas de *software*, *frameworks*, *toolkits*, *linguagens de programação*. Um **cookbook** é composto de **capítulos**, sendo que cada um está relacionado a um assunto ou tema da API alvo do cookbook. Cada capítulo é preenchido com uma série de **receitas**. Cada receita apresenta um cenário e instruções sobre como implementá-lo utilizando a API em questão. Cada receita contida em um capítulo deve estar relacionada com o tema do capítulo. **Os problemas tratados pelas receitas de um cookbook são questões práticas, ou seja, cenários que um desenvolvedor pode encontrar no seu dia-a-dia de trabalho. As receitas também são chamadas de "How-To"s, pois contêm instruções sobre "como-fazer" uma tarefa computacional (exemplo: "como ordenar um vetor de inteiros em Java")**. Esse tipo de problema é diferente dos casos em que o desenvolvedor precisa de ajuda para corrigir um bug em código-fonte defeituoso ou necessita saber algo conceitual (exemplo: "como funcionam os ponteiros em C").

Na Figura 1 é mostrada a capa de um cookbook existente para a biblioteca de *java script* JQuery (esse e vários outros cookbooks podem ser adquiridos em livrarias):

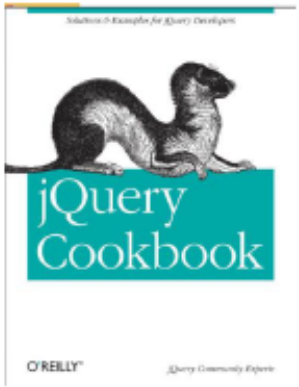


Figura 1 - Capa de um cookbook sobre JQuery

Neste cookbook sobre o JQuery, existem vários capítulos. A seguir exemplificamos os títulos de alguns deles:

- 2- Selecting Elements with JQuery;
- 6- Dimensions;
- 7- Effects;
- 8- Events.

Figura B.1: Documento de treinamento - página 1

Observe que cada capítulo possui um tema específico. Cada capítulo é preenchido com receitas. Por exemplo, os títulos das receitas contidas no capitulo "7- Effects" são mostrados na Figura 2.

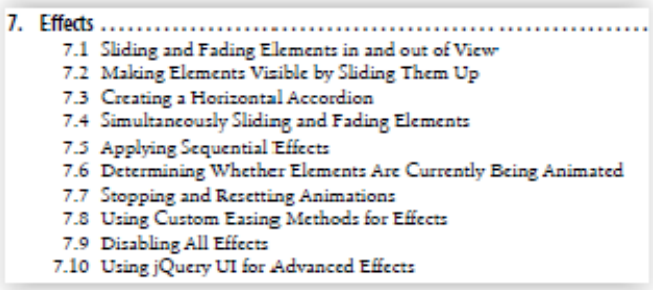
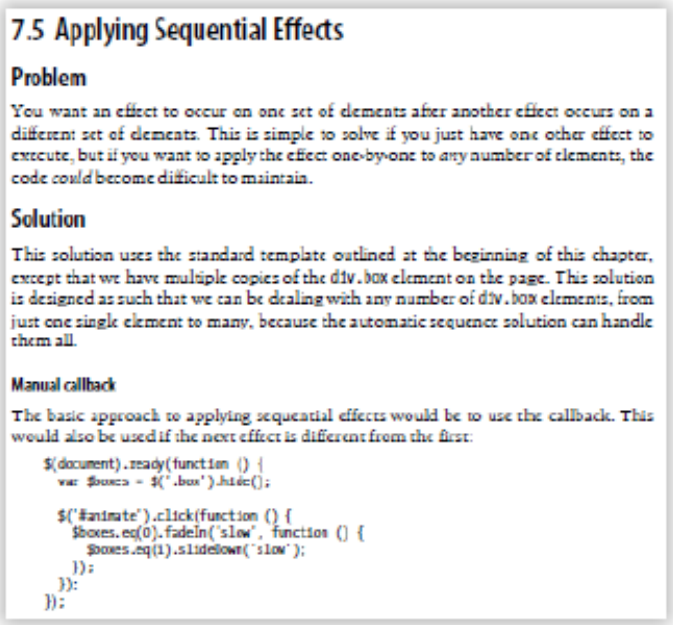


Figura 2 - Títulos das receitas presentes no capítulo 7 do cookbook sobre JQuery

A seguir, a título de ilustração, exemplificamos uma das receitas contidas no capítulo 7 deste cookbook (Figuras 3.a e 3.b). Observe que cada receita contém um problema (cenário) e uma solução que utiliza elementos da biblioteca JQuery, acompanhados de discussões/explicações.



7.5 Applying Sequential Effects

Problem

You want an effect to occur on one set of elements after another effect occurs on a different set of elements. This is simple to solve if you just have one other effect to execute, but if you want to apply the effect one-by-one to *any* number of elements, the code *could* become difficult to maintain.

Solution

This solution uses the standard template outlined at the beginning of this chapter, except that we have multiple copies of the `div.box` element on the page. This solution is designed as such that we can be dealing with any number of `div.box` elements, from just one single element to many, because the automatic sequence solution can handle them all.

Manual callback

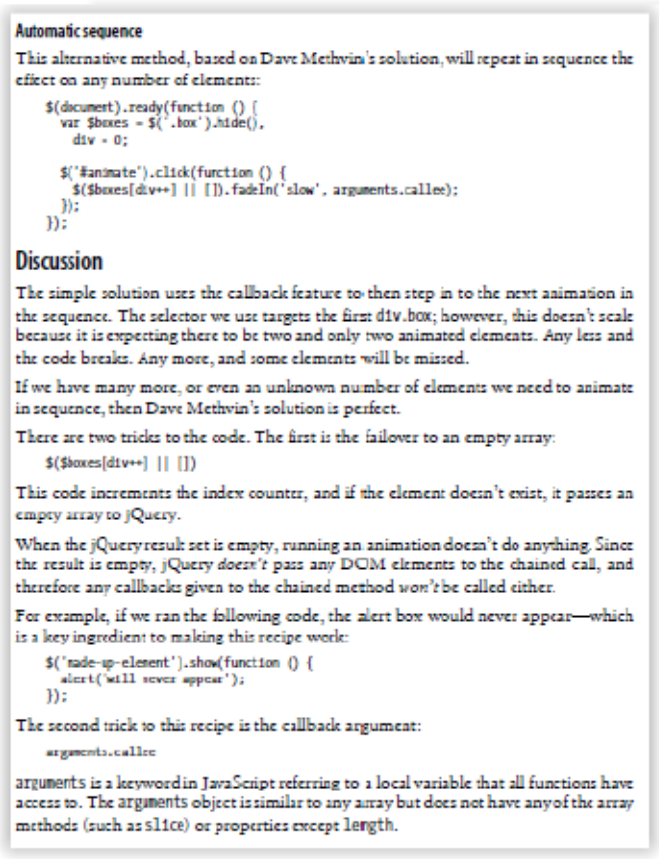
The basic approach to applying sequential effects would be to use the callback. This would also be used if the next effect is different from the first:

```
$(document).ready(function () {
  var $boxes = $('.box').hide();

  $('#animate').click(function () {
    $boxes.eq(0).fadeIn('slow', function () {
      $boxes.eq(1).slideDown('slow');
    });
  });
});
```

Figura 3.a - Exemplo de receita.

Figura B.2: Documento de treinamento - página 2



Automatic sequence

This alternative method, based on Dave Methvin's solution, will repeat in sequence the effect on any number of elements:

```
$(document).ready(function () {
  var $boxes = $('.box').hide(),
    div = 0;

  $('#animate').click(function () {
    $($boxes[div++] || []).fadeIn('slow', arguments.callee);
  });
});
```

Discussion

The simple solution uses the callback feature to then step in to the next animation in the sequence. The selector we use targets the first `div.box`; however, this doesn't scale because it is expecting there to be two and only two animated elements. Any less and the code breaks. Any more, and some elements will be missed.

If we have many more, or even an unknown number of elements we need to animate in sequence, then Dave Methvin's solution is perfect.

There are two tricks to the code. The first is the failover to an empty array:

```
$($boxes[div++] || [])
```

This code increments the index counter, and if the element doesn't exist, it passes an empty array to jQuery.

When the jQuery result set is empty, running an animation doesn't do anything. Since the result is empty, jQuery *doesn't* pass any DOM elements to the chained call, and therefore any callbacks given to the chained method *won't* be called either.

For example, if we ran the following code, the alert box would never appear—which is a key ingredient to making this recipe work:

```
$('made-up-element').show(function () {
  alert('will never appear');
});
```

The second trick to this recipe is the callback argument:

```
arguments.callee
```

`arguments` is a keyword in JavaScript referring to a local variable that all functions have access to. The `arguments` object is similar to any array but does not have any of the array methods (such as `slice`) or properties except `length`.

Figura 3.b (continuação da Figura 3.a): Exemplo de receita.

2.2 Cookbooks produzidos: Você irá responder um questionário com o objetivo de avaliar uma série de critérios quantitativos relacionados à **cookbooks**. Os cookbooks foram produzidos a partir de dados provenientes do site de perguntas e respostas **Stack Overflow**.

Foram construídos três cookbooks, um para cada uma das APIs descritas a seguir:

- LINQ (Language Integrated Query): é uma API para o *framework* Microsoft .NET, que adiciona capacidades de consulta às linguagens .NET. Essa API provê extensões curtas e de sintaxes expressivas para manipular dados.
- SWT é um *toolkit* para *widgets* da linguagem Java desenvolvido para prover acesso portável e eficiente aos elementos de interface gráfica dos sistemas operacionais.
- Qt é um *framework* multiplataforma para desenvolvimento de interfaces gráficas em C++. Com ele é possível desenvolver aplicativos e bibliotecas uma única vez e compilá-los para diversas plataformas sem que seja necessário alterar o código fonte.

Solicitamos que você não estude as APIs mencionadas, pois isso pode influenciar nos resultados obtidos pelas respostas do questionário.

A Figura 4 mostra a estrutura de capítulos de um cookbook construído para a API Swing. Neste caso, existem oito capítulos, cada um identificado por um código (exemplo: O código do primeiro capítulo é "Chapter 0"). O título de cada capítulo é representado por 5 termos

Figura B.3: Documento de treinamento - página 3

(palavras-chave). Por exemplo, o título do "Chapter 0" é composto pelos termos "model", "gui", "view", "control" e "data".

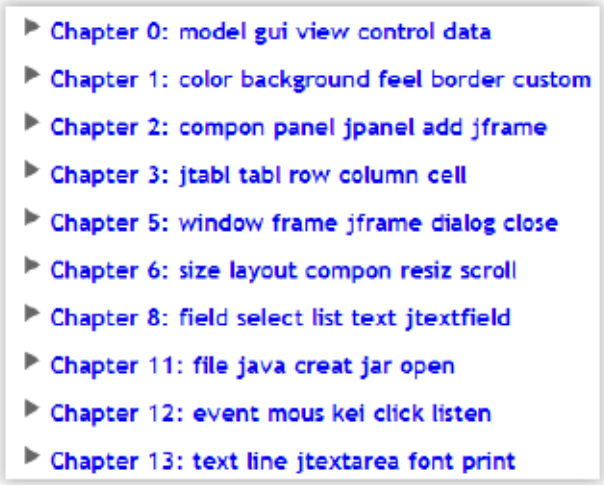


Figura 4 - Estrutura de capítulos de um cookbook produzido para a API Swing

Dentro de cada capítulo existem três ou mais receitas. A Figura 5 mostra o título das receitas contidas no "Chapter 3" do cookbook construído para o Swing.

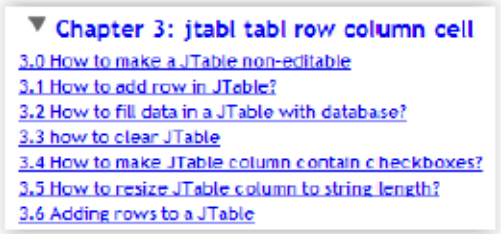


Figura 5 - Títulos das receitas presentes no capítulo 3

A Figura 6 exemplifica uma receita contida dentro do capítulo 3 do cookbook sobre Swing. Esta receita é identificada pelo código "3.5".

2.3 Critérios de Avaliação: Nessa subseção explicamos os critérios usados na avaliação dos cookbooks. No questionário a ser preenchido, você irá responder a várias questões em que serão solicitados que você dê notas para os critérios explicados adiante.

2.3.1 Adequação para ser parte de um cookbook : Este critério tem o objetivo de avaliar para cada receita, se a mesma é adequada para estar presente no cookbook em que ela se encontra. Conforme explicado anteriormente, receitas devem estar no formato "How-To", ou seja, apresentar um cenário e fornecer instruções sobre como resolvê-lo utilizando elementos (classes, métodos, etc.) da API alvo do cookbook.

Os valores para esse critério variam de 2 a -2:

- (2) Totalmente Adequada;
- (1) Parcialmente Adequada;
- (0) Não soube opinar;

Figura B.4: Documento de treinamento - página 4

- (-2) Totalmente inadequada.

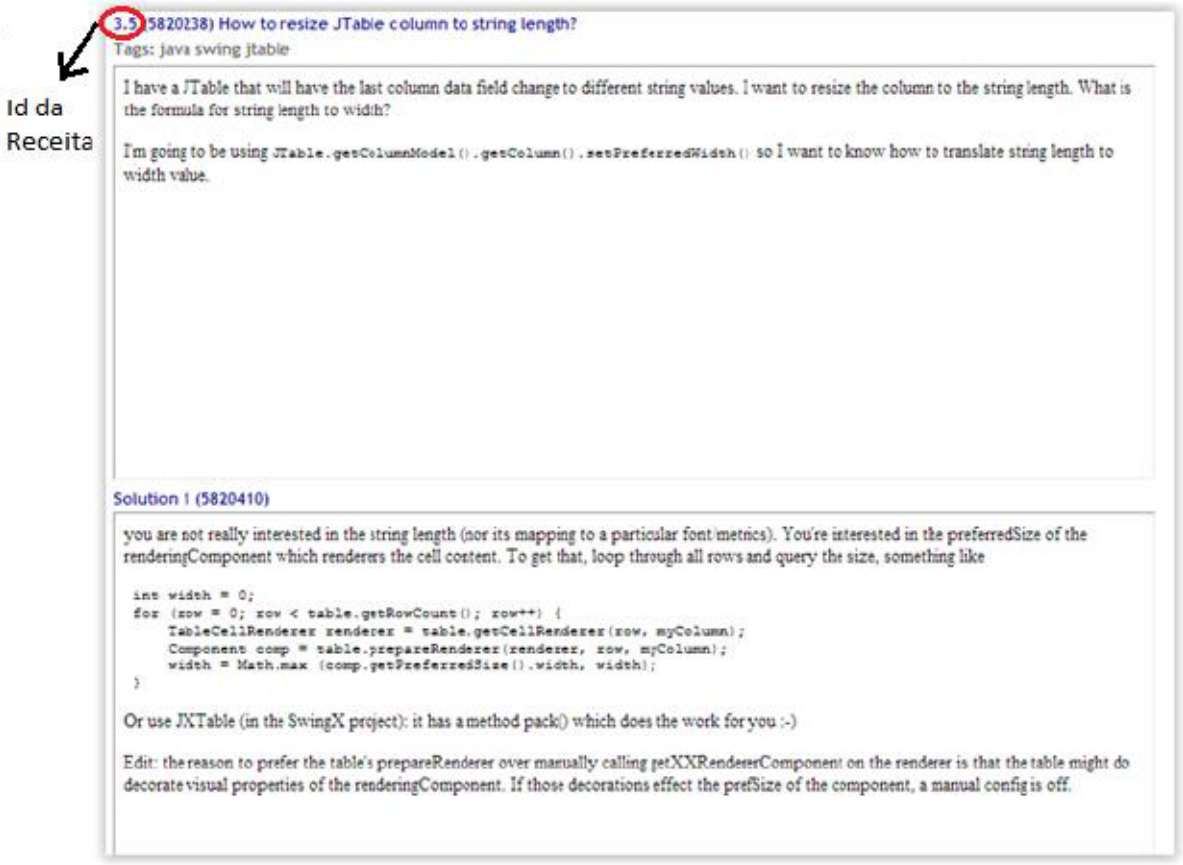


Figura 6 - Exemplo de Receita contida no capítulo 3.

Para dar nota a este critério você deverá ler toda a receita.

2.3.2 Auto-contenção: Este critério tem o objetivo de avaliar se as informações contidas na receita estão auto-contidas. Muitas receitas possuem links para fontes externas (exemplo: site da documentação oficial da API, *blogs*, fóruns). Apesar desse tipo de conteúdo ser benéfico no sentido servir como um complemento a solução apresentada no cookbook, não é desejado que a solução esteja presente apenas nesta fonte externa, pois não existe nenhuma garantia que as soluções apresentadas nestas fontes fiquem disponíveis para sempre (um dia esses sites podem "sair do ar"). Assim, é importante que as informações contidas em um cookbook estejam auto-contidas, ou seja, mesmo que existam referências a sites externos, as informações presentes no texto da receita devem ser suficientes para o entendimento do problema e solução lá apresentados. Os valores para esse critério variam de 2 a -2 e são explicados a seguir.

- (2) Totalmente auto-contida: É possível entender completamente o cenário e solução apresentados na receita apenas considerando-se o conteúdo textual da receita (sem acessar os possíveis sites referenciados por ela via links, caso eles existam);

Figura B.5: Documento de treinamento - página 5

- (1) Parcialmente auto-contida: É possível entender a maior parte do cenário e solução apresentados na receita apenas considerando-se o conteúdo textual da receita (sem acessar os possíveis sites referenciados por ela via links, caso eles existam);
- (0) Não soube opinar;
- (-1) Parcialmente não auto-contida: Para entender a maior parte do cenário e solução apresentados na receita é preciso acessar o conteúdo externo por ela referenciado (exemplos: sites externos);
- (-2) Totalmente não auto-contida: As informações contidas na receita são completamente dependentes de fontes externas referenciadas na receita (exemplo: links para outros sites).

2.3.3 Reprodutibilidade dos códigos-fontes: Este critério tem o objetivo de avaliar se o(s) trecho(s) de códigos-fonte presente(s) em cada receita podem ser diretamente compilados e executados usando um compilador ou um ambiente de programação como um IDE (exemplo: Eclipse). Os valores para este critério variam de 2 a -2 e são explicados a seguir.

- (2) O(s) trecho(s) pode(m) ser compilado(s) e executado(s) sem nenhuma alteração;
- (1) O(s) trecho(s) pode(m) ser compilado(s) e executado(s) depois de pequena(s) alteração(ões);
- (0) Não soube opinar;
- (-1) O(s) trecho(s) pode(m) ser compilado(s) e executado(s), porém para isso seria(m) necessária(s) alteração(ões) de médio a grande porte;
- (-2) Não é possível executar o(s) trecho(s) de código-fonte presente na receita ou ela não possui código-fonte.

2.3.4 Relacionamento com o capítulo: Conforme explicado, cada receita está contida em um capítulo. O título de cada capítulo é representado por 5 palavras-chave. O tema de cada receita deve estar relacionado com as palavras-chave representativas de seu capítulo. O objetivo deste critério é avaliar o relacionamento de cada receita com as 5 palavras-chave de seu capítulo. Os valores para este critério variam de 2 a -2 e são explicados a seguir.

- (2) Totalmente Relacionada: A receita está relacionada com três ou mais termos presentes no título do capítulo;
- (1) Parcialmente Relacionada: A receita está relacionada com dois termos presentes no título do capítulo;
- (0) Não soube opinar;
- (-1) Parcialmente não-relacionada: A receita está relacionada com apenas um dos termos presentes no título do capítulo;
- (-2) Totalmente não-relacionada: A receita não está relacionada com nenhum dos termos presentes no título do capítulo.

2.3.5 Semântica dos Capítulos: Este critério tem o objetivo de avaliar se é possível derivar um significado para o título do capítulo a partir de seus cinco termos representativos. A nota para

Figura B.6: Documento de treinamento - página 6

esse critério deverá estar na escala de 2 a -2. Cada valor desta escala de notas é explicado a seguir.

- (2) É possível atribuir um único significado (tema) preciso ao capítulo. Todos os cinco termos do título estão relacionados a este tema;
- (1) É possível atribuir um significado ao capítulo, porém algum(s) dos cinco termos não está(ão) relacionados ao tema encontrado;
- (0) Não soube opinar ou identificou mais de um tema;
- (-1) É possível perceber algum relacionamento entre os termos contidos no título, mas você não consegue atribuir um significado (tema) para este título;
- (-2) Não é possível perceber nenhum tipo de relacionamento entre os termos contidos no título.

Para dar nota a este critério você deverá apenas ler os títulos representativos de cada capítulo (não deverá considerar as receitas que estão dentro do capítulo). Para os capítulos avaliados com nota 2 ou 1, você deverá também escrever (em um campo destinado para isso) uma frase para descrever o significado por você encontrado para o capítulo, ou seja, você deverá nomear o capítulo.

3. Instruções Gerais

a) A URL de um questionário será enviada para você via e-mail. Para preenchê-lo, você deverá abrir em um navegador as versões eletrônicas dos cookbooks, cujas URLs serão divulgadas no mesmo e-mail;

b) O questionário consiste de um conjunto de questões (a maioria de múltipla escolha). Na maior parte das questões você será requerido a dar nota aos critérios explicados neste documento para receitas ou capítulos (os quatro primeiros critérios explicados neste documento são aplicados às receitas, e o quinto aos capítulos).

c) Além de dar nota aos critérios, pode-se opcionalmente, fazer alguma observação para justificar sua nota (haverá um campo para isso em cada questão);

d) Seja sincero no preenchimento do questionário: não fique apreensivo ao dar notas "ruins" para o que você achou ruim. A sua sinceridade é fundamental pra o sucesso da pesquisa, mesmo avaliando com notas ruins quando achar necessário;

e) Após o envio do questionário, você terá 7 dias para completá-lo;

f) Para os critérios avaliados com notas 1 (parcialmente positivo) e -1 (parcialmente negativo) encorajamos você a justificar o motivo da nota (para os outros valores, você também poderá justificar quando achar necessário).

Figura B.7: Documento de treinamento - página 7

B.2 Perguntas Relativas ao Documento de Treinamento

AS Figuras B.8 e B.9 mostram as perguntas realizadas aos participantes do experimento para verificar se eles estudaram e compreenderam o documento de treinamento.

Perguntas de Controle sobre o Documento de Treinamento

1. Dados do Participante

a) Nome completo:
b) Endereço:
c) E-mail:
d) Telefone(s):

2. Objetivo

O objetivo deste documento é fornecer ao participante da pesquisa algumas perguntas para avaliar se o mesmo entendeu os conceitos explicados no "Documento de Treinamento".

3. Perguntas

1- O que é um cookbook?
   a. ( ) Um livro de receitas culinárias;
   b. ( ) Um tipo de documentação para APIs (Application Program Interfaces) que contém um conjuntos de *JavaDocs*;
   c. ( ) Um tipo de documentação para APIs (Application Program Interfaces) orientado a receitas;
   d. ( ) Termo usado como sinônimo a "tutorial".

2- No contexto de um cookbook, o que é uma receita?
   a. ( ) Explicações sobre como resolver um bug em um código-fonte;
   b. ( ) Um cenário, seguido de instruções sobre como resolvê-lo. Os problemas tratados pelas receitas são os que geralmente um desenvolvedor encontra no seu dia-a-dia de trabalho;
   c. ( ) Um pergunta conceitual sobre uma linguagem de programação, acompanhada de resposta sobre a mesma;
   d. ( ) Um conjunto de bibliotecas de software voltado ao desenvolvimento de software para o ramo da culinária;

3- Marque V para verdadeiro e F para falso:
   a. ( ) Um cookbook é organizado em capítulos;
   b. ( ) Cada capítulo de um cookbook deve possuir um tema bem definido;
   c. ( ) Cada capítulo de um cookbook deve estar relacionado à vários temas;
   d. ( ) As receitas de um capítulo devem estar relacionadas ao tema do capítulo;
   e. ( ) Em um cookbook, podem existir capítulos vazios (sem receitas).

4- O critério "Adequação para ser parte de um cookbook" tem o objetivo de avaliar se:
   a. ( ) Uma receita possui links para fontes externas;
   b. ( ) Uma receita é adequada para estar no cookbook onde se encontra. Receitas devem estar no formato "How-To", ou seja, apresentar um cenário e fornecer

Figura B.8: Perguntas sobre o documento de treinamento - página 1

instruções sobre como resolvê-lo utilizando elementos (classes, métodos, etc.) da API alvo do cookbook;

c. ( ) Um capítulo possui sentido bem definido;
d. ( ) O cookbook é auto-contido.

5- O critério "Auto-contenção " tem o objetivo de avaliar se:
   a. ( ) É possível entender o cenário e solução apresentados na receita apenas considerando-se o conteúdo textual da receita;
   b. ( ) Os links para sites externos presentes na receita estão "quebrados";
   c. ( ) A solução apresentada na receita não é muito grande;
   d. ( ) A receita possui código-fonte.

6- O critério "Reprodutibilidade dos códigos-fontes:" tem o objetivo de avaliar se:
   a. ( ) A receita apresenta código-fonte;
   b. ( ) O(s) trecho(s) de código-fonte presente(s) na receita podem ser compilados e executados;
   c. ( ) O(s) trecho(s) de código-fonte presentes na receita possuem variáveis com nomes auto-explicativos;
   d. ( ) O texto presente na receita é de fácil leitura e entendimento.

7- O critério "Relacionamento com o capítulo" tem o objetivo de avaliar se:
   a. ( ) A receita está relacionada com os termos representativos de seu capítulo;
   b. ( ) Os termos representativos do capítulo estão relacionados entre si;
   c. ( ) É possível "nomear" o capítulo a partir dos termos representativos de seu título;
   d. ( ) Os capítulos estão relacionados entre si.

8- O critério "Semântica dos Capítulos" tem o objetivo de avaliar se:
   a. ( ) É possível perceber um tema para o capítulo a partir da análise dos termos representativos do seu título;
   b. ( ) A receita está semanticamente relacionada com a API do cookbook;
   c. ( ) A receita está possui explicações em linguagem natural;
   d. ( ) O cookbook está auto-contido.

9- Qual seu nível para leitura e entendimento de textos técnicos em inglês:
   a. ( ) Avançado;
   b. ( ) Intermediário;
   c. ( ) Iniciante;
   d. ( ) Nenhum.

Figura B.9: Perguntas sobre o documento de treinamento - página 2

Apêndice C

Dados sobre a Amostragem

C.1 Capítulos e Receitas Selecionados pela Amostragem

Os capítulos selecionados pela amostragem para a avaliação do *cookbook* **SWT-*Cookbook***[1] são:

- Chapter 2: tabl column row cell tableview;
- Chapter 4: tabl column row cell tableview;
- Chapter 11: file jar eclips icon plugin;
- Chapter 12: wizard dispos page tab file.

Os capítulos selecionados pela amostragem para a avaliação do *cookbook* **SWT-Controlled-*Cookbook***[2] são:

- Chapter 2: tabl column row cell tableview;
- Chapter 4: tabl column row cell tableview;
- Chapter 11: file jar eclips icon plugin;
- Chapter 12: wizard dispos page tab file;
- Chapter 3: pdf press thread eclipse wizard.

Os capítulos selecionados pela amostragem para a avaliação do *cookbook* **LINQ-*Cookbook***[3] são:

- Chapter 1: type properti anonym creat anonym_type;

[1] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção swt
[2] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção swt_fake
[3] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção linq

- Chapter 8: perform loop iter call enumer;
- Chapter 10: perform loop iter call enumer;
- Chapter 11: perform loop iter call enumer.

Os capítulos selecionados pela amostragem para a avaliação do *cookbook* **LINQ-Controlled-*Cookbook***[4] são:

- Chapter 1: type properti anonym creat anonym_type;
- Chapter 8: perform loop iter call enumer;
- Chapter 10: perform loop iter call enumer;
- Chapter 11: perform loop iter call enumer;
- Chapter 6: group color xml anonym_type loop.

Os capítulos selecionados pela amostragem para a avaliação do *cookbook* **QT-*Cookbook***[5] são:

- Chapter 0: string qstring pointer type data;
- Chapter 2: window platform linux mac process;
- Chapter 7: creat http call document find;
- Chapter 14: file compil build instal librari.

Os capítulos selecionados pela amostragem para a avaliação do *cookbook* **QT-Controlled-*Cookbook***[6] são:

- Chapter 0: string qstring pointer type data;
- Chapter 2: window platform linux mac process;
- Chapter 7: creat http call document find;
- Chapter 14: file compil build instal librari;
- Chapter 1: row imag compil slot event.

As receitas selecionados pela amostragem para a avaliação do *cookbook* **SWT-*Cookbook*** são: 7.2, 13.2, 2.1, 4.6, 10.0, 1.0, 8.3, 11.3, 9.3, 5.3, 12.2, 14.3, 4.5, 11.1, 14.1, 12.1.

As receitas selecionados pela amostragem para a avaliação do *cookbook* **SWT-Controlled-*Cookbook*** são: 7.2, 13.2, 2.1, 4.6, 10.0, 1.0, 8.3, 11.3, 9.3, 5.3, 12.2, 14.3, 11.5, 5.4, 1.6, 2.8.

[4] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção linq_fake
[5] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção qt
[6] http://lascam.facom.ufu.br:8080/QAWeb_dis/ -> opção qt_fake

As receitas selecionados pela amostragem para a avaliação do *cookbook* **LINQ-*Cookbook*** são: 7.3, 1.3, 10.17, 12.2, 8.5, 5.2, 11.1, 0.6, 2.3, 0.4, 12.1, 10.28, 7.11.

As receitas selecionados pela amostragem para a avaliação do *cookbook* **LINQ-Controlled-*Cookbook*** são: 7.3, 1.3, 10.17, 12.2, 8.5, 5.2, 11.1, 0.6, 2.3, 10.43, 5.3, 11.5, 11.6.

As receitas selecionados pela amostragem para a avaliação do *cookbook* **QT-*Cookbook*** são: 14.13, 7.1, 3.1, 9.0, 13.3, 0.5, 12.1, 8.1, 10.0, 6.0, 5.1, 2.1, 2.2, 0.18, 8.4, 14.16.

As receitas selecionados pela amostragem para a avaliação do *cookbook* **QT-Controlled-*Cookbook*** são: 14.13, 7.1, 3.1, 9.0, 13.3, 0.5, 12.1, 8.1, 10.0, 6.0, 5.1, 2.1, 2.5, 6.6, 7.6, 8.9.

Apêndice D

Respostas dos Avaliadores Quanto ao Uso dos *Cookbooks* para aprendizado de APIs

D.1 Opiniões dos Avalidores

As opiniões dos participantes quanto à utilidade dos *cookbooks* **SWT-*Cookbook***, **LINQ-*Cookbook*** e **QT-*Cookbook*** são mostradas nas Tabelas D.1, D.2 e D.3 respectivamente.

Tabela D.1: Opiniões acerca do uso do *cookbook* **SWT-*Cookbook*** para aprendizado

Sujeito	Usaria o *cookbook* para aprender sobre SWT?
1	Não, pois eu acredito que para iniciantes há a necessidade de (uma sequência de) tópicos mais simples, para começar com um "hello world"em SWT mesmo. Já para usuários SWT (que é o meu caso), o *cookbook* é interessante, até porque os nomes dos capítulos são formados por palavras-chave, e usuários SWT conseguem encontrar mais facilmente o que procuram.
3	Sim. Fazendo clonagem de código e executando-os em minhas aplicações.
6	Não. Usaria como um guia de referência para solução de alguns problemas.
7	Usaria quando resolver algum problema pontual e específico, isto é, como consulta rápida para tirar uma dúvida. O formato de um *cookbook* não me serviria de guia para aprender algo novo, como SWT.
9	Usaria o *cookbook*, mas para tirar dúvidas, como estudo seria pouco utilizado.
11	Sim, mas com algumas ressalvas. Eu o usaria mais como uma consulta rápida para algum problema específico, do que de fato para aprender sobre a API. Isso porque o tema dos capítulos não é imediato, e precisa ser derivado a partir da conexão das palavras-chave. Além disso, os problemas apresentados (ou seja, as perguntas feitas pelos usuários nos fóruns) podem não estar claros, sendo difícil para alguém que não conhece a API entender o que realmente está colocado.
12	Sim, para aprendizado.
13	Sim. Usaria o mesmo para sanar as dúvidas que surgissem.
14	Sim. Utilizaria de acordo com a demanda. Por isso a importância do nome dos capítulos estarem bem definidos.
15	Sim, diversas informações básicas foram abordadas no *cookbook*. Informações essas necessárias para qualquer um que deseje iniciar um aprendizado na API.

Tabela D.2: Opiniões acerca do uso do *cookbook* **LINQ-*Cookbook*** para aprendizado

Sujeito	Usaria o *cookbook* para aprender sobre LINQ?
1	Não consigo responder.
4	Sim. Eu usaria, pois este coookbook apresenta um conjunto de respostas para dúvidas comuns ao uso da API. Com exemplos de código que se não resolvem o problema, podem direcionar a solução. Além do mais, eu preferiria usar o *cookbook* à buscar as respostas no Stack Overflow, pois no *cookbook* as receitas estão separadas por tópicos (temas) que a mim pareceram, na maioria dos casos, intuitivos, facilitando a busca e agrupando interesses.
5	Sim, se algumas receitas fossem trocadas de lugar. Tem umas que me parecem estar fora do local.
6	Não. Usaria como um guia de referência para solução de alguns problemas.
7	Também a usaria como guia de consulta rápida, quando tivesse dúvidas sobre como fazer algo específico.
8	Sim. Seria para os casos em que tenho dúvidas em algum ponto de implementação e nesse caso verifico se já existe alguma resposta no *Cookbook*.
10	De certa forma sim. Não para aprender, mas para tirar dúvidas básicas durante o meu aprendizado.
13	Sim, somente para sanar dúvidas não encontradas de forma eficiente em sites.
16	Eu usaria o *cookbook* de LINQ para aprender sobre a API se já tivesse algum conhecimento, ou seja, para tirar duvidas. Ele funcionária como um tutorial focado em temas específicos. Pois o *cookbook* não segue uma ordem cronológica necessária para aprender uma API. Levando em conta que este não é o objetivo de um *cookbook* ele cumpre o papel corretamente.

Tabela D.3: Opiniões acerca do uso do *cookbook* **QT-*Cookbook*** para aprendizado

Sujeito	Usaria o *cookbook* para aprender sobre QT?
3	Sim. Usaria como uma possível fonte de recursos de código disponível que pudesse ser útil na construção de minhas aplicações. Entretanto, em particular como não tenho nenhum conhecimento sobre QT provavelmente recorria também à outras fontes externas.
4	Eu usaria, pois este coookbook apresenta um conjunto de respostas para dúvidas comuns ao uso da API. Com exemplos de código que se não resolvem o problema, podem direcionar a solução. Além do mais, eu preferiria usar o *cookbook* à buscar as respostas no Stack Overflow, pois no *cookbook* as receitas estão separadas por tópicos (temas) que a mim pareceram, na maioria dos casos, intuitivos, facilitando a busca e agrupando interesses.
5	Sim e se quiser lançar uma versão dele pra pyqt eu agredeço hehehhee
8	Sim. Mas como uma fonte de consulta nos casos de dúvidas em alguma implementação. Conta com vários temas que são de grande ajuda para o dia a dia.
9	Usaria o *cookbook*, mas para tirar dúvidas, como estudo seria pouco utilizado.
10	Não, pois é muito técnico, serviria mais para quem já tem conhecimento do assunto e se deparou com algum problema.
11	Sim, mas com algumas ressalvas. Eu o usaria mais como uma consulta rápida para algum problema específico, do que de fato para aprender sobre a API. Isso porque o tema dos capítulos não é imediato, e precisa ser derivado a partir da conexão das palavras-chave. Além disso, os problemas apresentados (ou seja, as perguntas feitas pelos usuários nos fóruns) podem não estar claros, sendo difícil para alguém que não conhece a API entender o que realmente está colocado.
12	Sim, mas faltaria muitas outras coisas a serem complementas, como pequenos tutorias para uma melhor aprendizagem.
14	Se eu já tivesse competência sobre QT, utilizaria sim.
15	Sim, diversas informações básicas foram abordadas no *cookbook*. Informações essas necessárias para qualquer um que deseje iniciar um aprendizado na API.
16	Eu usuaria o *cookbook* de QT apenas para consulta de duvidas específicas. Para que ele possa ser útil para aprender uma tecnologia do inicio ao fim deveria estruturar as questões baseando-se em um índice de algum livro conhecido da tecnologia em questão.