6
Conclusão

6.1.
Considerações finais

Um dos objetivos da Engenharia de Software é construir sistemas de software fidedignos, isto é, sistemas que sejam dignos da confiança de seus usuários. Para alcançar tal objetivo, é necessário que os sistemas de software provejam seus serviços corretamente sob diferentes condições de operação. Mesmo em condições excepcionais de operação, os sistemas de software devem ser capazes de tolerar tais condições e prover seus serviços corretamente. Ou ao menos serem capazes de comportar-se aceitavelmente. Atualmente os mecanismos de tratamento de exceções são os modelos mais comumente usados em linguagens de programação para a detecção de ocorrências de exceções e para a estruturação do fluxo excepcional de sistemas de software. O sucesso dos mecanismos de tratamento de exceções pode ser atestado pela sua adoção em diversas linguagens de programação contemporâneas e de largo uso comercial, como Java e C#, por exemplo. Mas ainda que a implementação do código de tratamento de exceções de um sistema de software seja de suma importância para a realização de sua fidedignidade, esta não é, no entanto, uma atividade simples de ser realizada.

Estudos anteriores mostram que desenvolvedores em geral tendem a ignorar o tratamento de exceções, considerando-o uma tarefa secundária no ciclo de desenvolvimento. Outros estudos mostram que o código de tratamento de exceções comumente implementado em sistemas reais da indústria é de má qualidade. A falta de qualidade do tratamento de exceções está relacionado com problemas de desempenho, falhas recorrentes e, em casos extremos, degradação do projeto de software de um sistema. Percebe-se assim que a abordagem de ignorar o tratamento de exceções adotada por muitos desenvolvedores pode ter consequências danosas à fidedignidade de sistemas de software.

Na primeira parte da pesquisa realizada no contexto desta dissertação, a qualidade do código de tratamento de exceções de *frameworks* e aplicações de

software foi avaliada (Seção 2.3). Esta primeira avaliação corroborou estudos anteriores que já apontavam a má qualidade do código de tratamento de exceções. Em especial, verificou-se com frequência a ineficácia dos tratadores implementados. Além disso, esta primeira avaliação mostrou que poucos esforços são alocados em termos de melhorar o tratamento de exceções durante a evolução de um sistema. Na maioria dos cenários de evolução analisados, nenhum esforço para melhorar a qualidade dos tratadores foi identificado. Tal falta de melhorias durante a evolução deve-se muito à dificuldade em se modificar código de tratamento de exceções, dado o seu impacto global no sistema. Desta forma, deixar para realizar o tratamento de exceções corretamente apenas em versões futuras de um sistema pode ser bastante custoso. Em alguns casos, pode ser até danoso. O projeto inadequado do tratamento de exceções pode arruinar o projeto de software de um sistema durante sua evolução. Advoga-se assim a necessidade de implementar tratamento de exceções corretamente desde as primeiras versões de um sistema.

Ainda na primeira avaliação realizada verificou-se a inadequação das ferramentas já propostas para auxiliar desenvolvedores de software a lidar com código de tratamento de exceções. Essas ferramentas propostas falham principalmente em auxiliar desenvolvedores a implementar de maneira mais adequada o tratamento de exceções de suas aplicações desde as primeiras versões de um sistema. É neste contexto que esta dissertação faz sua principal contribuição.

No contexto desta dissertação foi proposto um sistema de recomendação para código de tratamento de exceções (Capítulo 3). O objetivo do sistema de recomendação é prover exemplos de código implementando tratadores de exceções de modo a auxiliar desenvolvedores no processo de aprendizado. Não é objetivo do sistema de recomendação proposto prover exemplos de código que sejam reutilizáveis *ipsis litteris.* É esperado que os desenvolvedores usem os exemplos de código como forma de aprender novas e melhores formas de implementar o tratamento de exceções de suas aplicações. Com isso, espera-se, de maneira otimista, que estes desenvolvedores passem a implementar código de tratamento de exceções de melhor qualidade desde o princípio.

O sistema de recomendação proposto usa um conjunto de heurísticas para realizar buscas em um repositório de exemplos. Tais exemplos de código são

extraídos de projetos de software de código aberto disponíveis em repositórios públicos da Internet. As heurísticas extraem informações estruturais do código em que o desenvolvedor está trabalhando e buscam por exemplos que compartilhem similaridades estruturais. O objetivo de se buscar por similaridades é tentar encontrar exemplos de código tratando exceções em contextos parecidos com o contexto em que o desenvolvedor está trabalhando. Os candidatos do repositório de exemplos mais relevantes são então recomendados aos desenvolvedores.

Também foi implementado um primeiro protótipo do sistema a fim de avaliar o conceito proposto (Capítulo 4). O sistema de recomendação foi implementado como um plugin do Eclipse, para que os desenvolvedores possam desfrutar de seus serviços sem a necessidade de sair do seu ambiente de desenvolvimento. Por fim, foram realizadas avaliações do sistema de recomendação implementado. As avaliações realizadas foram de caráter mais exploratório. Esta estratégia empírica foi escolhida por ser a mais apropriada quando não se possui uma ferramenta com interface adequada, um requisito essencial para avaliações baseadas em experimentos controlados (KITCHENHAM et al, 2002). As avaliações realizadas (Capítulo 5) foram divididas em duas categorias: avaliações preliminares (Seção 5.1), realizadas já durante a implementação do sistema, e avaliações de relevância (Seção 5.2), realizadas após terminada a implementação.

As avaliações preliminares tiveram como objetivo primário prover retroalimentação rápida através de experimentos analíticos. Tais experimentos foram realizados com o auxílio de programas executáveis, que podiam ser executados sem altos custos de execução durante o processo de desenvolvimento. As avaliações preliminares proveram informações sobre a suficiência dos fatos estruturais representados pelo modelo de dados e sobre a influência do ajuste de pesos na capacidade das heurísticas em detectar similaridades estruturais. As avaliações mostraram que os fatos estruturais usados são capazes de detectar similaridades estruturais entre fragmentos de código. Além disso, também mostraram que o ajuste de pesos pode aumentar a capacidade das heurísticas em detectar similaridades estruturais. Formulou-se ainda a hipótese de que fatos estruturais ordinários, i.e., fatos estruturais bastante comuns no repositório de exemplos, não têm grande poder de realçar similaridades estruturais entre candidatos. Além disso, em alguns casos os fatos estruturais ordinários até

atrapalharam a precisão das heurísticas ao realçar similaridades com diversos candidatos.

Já as avaliações de relevância das informações recomendadas tiveram como objetivo verificar se as heurísticas são capazes de recomendar fragmentos de código contendo informações relevantes. As avaliações de relevância mostraram que as heurísticas conseguiram recomendar informações relevantes em todos os cenários testados. Nas avaliações de relevância também foi explorada a relação entre a relevância das informações e o tamanho e proveniência do conjunto de fatos estruturais. Alguns fenômenos observados motivaram a formulação de duas hipóteses que ainda não foram validadas. A primeira hipótese afirma que quanto maior o conjunto de fatos provenientes do *framework*, mais relevantes serão as recomendações. Já a segunda hipótese afirma que quanto mais raro for um fato estrutural no repositório de exemplos, maior será a chance de ele ressaltar um candidato com informações relevantes. Verificou-se ainda que em muitos casos os exemplos recomendados continham métodos muito grandes, com 300 linhas ou mais. Este fato recorrente aponta para melhorias futuras em termos de: (i) prover uma melhor apresentação para os exemplos recomendados, (ii) refinar a função de pontuação, levando em conta o tamanho do exemplo recomendado e (iii) refinar o mecanismo de extração de exemplos, desconsiderando métodos muito grandes.

Ainda que o tamanho da amostra não permita afirmar categoricamente que "sim, as heurísticas recomendam informações relevantes", as avaliações realizadas dão um bom indício para isto. Os resultados das avaliações levam a crer que a recomendação de exemplos de código pode ser um eficiente recurso no auxílio a desenvolvedores em tarefas relacionadas a implementação do tratamento de exceções em sistemas de software. Principalmente desenvolvedores menos experientes podem se beneficiar das recomendações.

6.2. Contribuições

Há muito se advoga sobre a importância do tratamento de exceções em sistemas de software. No entanto, ainda hoje percebe-se que desenvolvedores em geral têm dificuldade e relutam em implementar tratamento de exceções adequadamente em suas aplicação. Seja qual for o real motivo para este problema,

a verdade é que o tratamento de exceções faz parte do processo de desenvolvimento de software e deve ser bem realizado pelos desenvolvedores. Por este motivo, é necessário que os desenvolvedores parem de ignorá-lo e aprendam a realizá-lo de maneira adequada. Nesta dissertação, foi proposto um sistema de recomendação cujo objetivo é auxiliar desenvolvedores no aprendizado do tratamento de exceções através do provimento de exemplos concretos de código implementando tratamento de exceções. Neste contexto, as principais contribuições desta dissertação são:

1. Avaliação da qualidade do código do tratamento de exceções de *frameworks* e aplicações industriais durante períodos de evolução de dois ou mais anos.

   A avaliação da qualidade do código do tratamento de exceções de sistemas de software reais de uso na indústria mostrou alguns fenômenos interessantes que motivaram a construção do sistema de recomendação. Em particular, destaca-se a ineficácia dos tratadores de exceções implementados, bem como a falta de esforços em melhorar estes tratadores ao longo da evolução dos sistemas. Essa avaliação realizada nos estudos preliminares desta dissertação foi reconhecida pela comunidade científica de engenharia de software, da qual obteve-se aceitação para a publicação de dois artigos científicos (BARBOSA e GARCIA, 2010; BARBOSA e GARCIA, 2011). Essa avaliação foi apresentada no Capítulo 2.

2. Definição de estratégias heurísticas capazes de buscar e ranquear exemplos de código implementando código de tratamento de exceções.

   Estas heurísticas realizam suas buscas baseadas em informações estruturais extraídas do código em que o desenvolvedor está trabalhando. Elas são usadas para recomendar a desenvolvedores exemplos de código implementando tratamento de exceções. A definição das heurísticas de recomendação foi submetida para o *4th Workshop on Exception Handling – WEH'12*, evento que será realizado conjuntamente com o *34th International Conference on Software Engineering – ICSE'12*. Aguarda-se ainda o resultado da aceitação deste artigo submetido. As heurísticas foram apresentadas no Capítulo 3.

3. Implementação de um sistema de recomendação baseado nas estratégias heurísticas propostas.

Implementou-se um primeiro protótipo de um sistema de recomendação baseado nas heurísticas propostas. Para tanto, também foi necessária a construção de um repositório de exemplos concretos de código implementando tratamento de exceções. Tais exemplos concretos foram extraídos de projetos de software de código aberto disponíveis na Internet. A implementação do sistema de recomendação foi apresentada no Capítulo 4.

4. Avaliações qualitativa e quantitativa da precisão e da relevância das informações recomendadas pelas estratégias heurísticas.

A implementação das heurísticas de recomendação foram avaliadas qualitativamente, em termos de relevância das recomendações, e quantitativamente, em termos de precisão das similaridades estruturais capturadas pelo modelo de dados. Os resultados mostraram que as heurísticas de recomendação são capazes de encontrar exemplos estruturalmente similares e que contêm informações relevantes aos desenvolvedores em tarefas de implementação de tratamento de exceções. As avaliações da implementação das heurísticas foram apresentadas no Capítulo 5.

6.3. Trabalhos futuros

Os resultados obtidos e as contribuições apresentadas são apenas um primeiro esforço no sentido de prover um melhor aparato ferramental aos desenvolvedores em termos de auxílio à implementação de tratamento de exceções. Há ainda diversos quesitos a serem melhorados no sistema de recomendação em trabalhos futuros, dentre os quais enumeram-se:

1. Refinar as estratégias heurísticas.

As estratégias heurísticas baseiam suas buscas em um conjunto conciso de fatos estruturais. Em trabalhos futuros, pretende-se explorar o uso de outros fatos estruturais, como o identificador das variáveis usadas, o tipo da classe que a classe que contém a declaração do método está estendendo, ou se a declaração do método sobrescreve algum método da classe mãe.

2. Refinar a implementação.

O foco principal desta dissertação foi a definição e implementação das heurísticas de recomendação. Por este motivo, o módulo de interface gráfica acabou recebendo menor atenção durante a fase de implementação do sistema de recomendação. Sabe-se, no entanto, que este é um módulo importante para o sistema. Na próxima versão do sistema, pretende-se definir e implementar um módulo de interface gráfica mais adequado ao usuário final. Pretende-se, principalmente, definir e implementar um mecanismo de apresentação das recomendações capaz de atenuar alguns problemas já identificados durante as avaliações realizadas. Em especial, os problemas relacionados à apresentação de exemplos de código muito grandes.

O sistema de recomendação permite que os seus desenvolvedores-usuários ajustem os pesos das heurísticas de modo a refinar os resultados obtidos. Entretanto, os desenvolvedores-usuários não têm como prover explicitamente retroalimentação para o sistema de recomendação. Em trabalhos futuros, pretende-se permitir que o desenvolvedor proveja retroalimentação a respeito das recomendações, de modo que o sistema de recomendação ajuste-se de acordo com o perfil do usuário.

Uma funcionalidade bastante simples, mas que não foi contemplada neste primeiro protótipo, é permitir que o desenvolvedor inclua novos exemplos no repositório. Pretende-se atender esta demanda na próxima versão do sistema de recomendação.

3. Realizar outras avaliações.

As avaliações realizadas nesta dissertação foram realizadas com aplicações baseadas em um mesmo *framework,* o Eclipse. Ainda que as aplicações utilizadas sejam bastante heterogêneas, em trabalhos futuros pretende-se realizar avaliações com aplicações de outros nichos, como bibliotecas de código reutilizável, jogos, servidores web, dentre muitas outros. Pretende-se assim verificar se os resultados iniciais obtidos se replicam nestes outros nichos de aplicação.

Por fim, pretende-se realizar experimentos controlados com desenvolvedores tão logo se termine a implementação de um módulo de interface gráfica adequado ao usuário final. O objetivo do sistema de recomendação é auxiliar desenvolvedores em tarefas de implementação de código de tratamento de

exceções. Desta forma, o sistema de recomendação só será avaliado mais adequadamente quando desenvolvedores o utilizarem em cenários simulando cenários reais de desenvolvimento. Este é o principal objetivo a ser alcançado nos trabalhos futuros.