6
Conclusão e Trabalhos Futuros

Este capítulo pretende resumir o trabalho descrito nesta dissertação e apresentar as conclusões alcançadas durante o estudo, além de enumerar as contribuições do presente trabalho e delinear o modo como foi realizado. Por fim, esta seção também busca fazer indicações para futuras pesquisas.

6.1.Conclusões

Este trabalho pode ser dividido em duas partes principais que se complementam, sendo a primeira (i) o estudo exploratório; e a segunda (ii) o desenvolvimento da ferramenta *SAFE-JBossAOP* (*Static Analysis for the Flow of the Exceptions for JBoos AOP*), uma ferramenta de análise de fluxo de exceção para programas que utilizem *JBoss AOP*.

Realizamos um estudo exploratório para avaliar o impacto dos aspectos sobre tratamento de exceções de alguns programas. Neste estudo foram selecionados três sistemas implementados na linguagem Java e que utilizam o *JBoss AOP* através do *framework EJB*, são eles: (i) *CMS* (*Conference Management System*), um sistema para gerenciamento de conferências; (ii) *COS-HMS*, um sistema para gerenciamento hospitalar; (iii) *WordNet*, um sistema para gerenciamento de palavras e seus relacionamentos utilizado como apoio a sistemas de processamento de linguagens.

Nos sistemas analisados, observamos um padrão de tratamento de exceção que consiste em: (i) capturar todas as exceções que herdam de `java.lang.Exception` na camada de interface gráfica, e (ii) apresentar uma mensagem genérica de erro ou encapsular a exceção em outra. Acreditamos que esta abordagem de tratamento de erro ocorre devido à quantidade de exceções de classes distintas e que são lançadas pelos aspectos e pelas bibliotecas utilizadas pela aplicação. Desta forma, os desenvolvedores optaram por capturar todas as exceções por subsunção e não tratar cada exceção individualmente, uma razão

para isto é que a maioria dos fluxos excepcionais ocorre devido a exceções não-checadas (ver Seção 4.3.1). Além disso, identificamos que aplicações que utilizam o *framework EJB* possuem pontos específicos para os aspectos interceptarem, e desta forma, facilitando o tratamento de exceções. Diferentemente das aplicações analisadas por Coelho (2008), nas quais as os aspectos interceptavam a aplicação em diversos pontos da arquitetura (ver Seção 5.2).

Para apoiar o estudo empírico descrito acima desenvolvemos a ferramenta *SAFE-JBossAOP* que foi construída a partir da ferramenta *SAFE* (COELHO, 2008). A nossa ferramenta analisa estaticamente aplicações e calcula os fluxos de exceções de programas *JBoss AOP*. O algoritmo de análise do fluxo de exceção, implementado na ferramenta, percorre uma representação do programa, gerada pelo *frameowork* SOOT, para descobrir: (i) a cada exceção que pode escapar de um aplicativo ou um método de aspectos, e (ii) o caminho de exceção que cada exceção percorre quando a exceção é lançada.

No processo de criação de uma ferramenta de análise estática a principal preocupação é determinar com precisão quais são os fluxos que ocorrem na aplicação analisada. Uma das técnicas utilizadas para determinar com precisão o fluxo na aplicação é *Point-to* (ANDERSEN, 1994), esta técnica permite inferir qual o tipo da instância das variáveis da aplicação, a partir das instruções construção de objetos (`new`). Contudo, os frameworks mais recentes para desenvolvimento de aplicações utilizam *containers* (ver Seção 2.3) e reflexão computacional para instanciar os objetos, e desta forma dificultando o processo de análise estática. Na nossa ferramenta utilizamos *Point-to* para melhorar a precisão das invocações de métodos analisadas, contudo, algumas bibliotecas precisam de informações externas para instanciar os objetos (ver Seção 0). Estas bibliotecas afetam a análise estática de forma negativa de tal forma algumas classes foram removidas da análise para evitar que caminhos impossíveis fossem computados pela ferramenta. Por outro lado, detectamos que a execução de aspectos, que utilizam informações do contexto de excecução, afeta negativamente a precisão da ferramenta (ver Seção 4.3.1.3).

6.2.Contribuições

As principais contribuições deste trabalho são as listadas e detalhadas abaixo:

- **Estudo Exploratório**. Este trabalho apresenta uma análise sistemática que investiga aplicações como um todo, ou seja, levando em consideração tanto os aspectos quanto as bibliotecas que compõem a aplicação, de forma a mostrar como os aspectos e suas bibliotecas afetam os fluxos de exceção dos programas (Capítulo 4).
- **SAFE-JBossAOP** uma ferramenta de análise de programas JBoss AOP. Este trabalho apresenta uma ferramenta de análise de fluxo de exceção em programas que utilizem o JBoss AOP (Capítulo 3).

6.3.Trabalhos Futuros

O presente trabalho dá margem a novos estudos acerca do tema, visto que existem pontos a serem explorados ou estudos baseados na ferramenta proposta na presente dissertação. Dentre eles, podemos citar alguns como descrito abaixo:

Armazenamento de informações do fluxo de exceção de bibliotecas em banco de dados – Guardar informações sobre o fluxo de exceção das bibliotecas em banco de dados evita que seja necessário analisar novamente uma biblioteca para mais de uma aplicação. Isso deve impactar positivamente o desempenho da nossa análise inter-procedural.

Integrar a ferramenta SAFE com uma IDE – O fluxo de exceção informação calculado pela SAFE em tempo de compilação pode ser integrado com um ambiente de desenvolvimento como o Eclipse como proposto por (SINHA *et alii*, 2004). Essa integração permitirá que o desenvolvedor navegue através das interfaces de exceção dos métodos durante desenvolvimento, isto pode ajudar a remover os defeitos que foram relatados pela ferramenta. Tal integração poderia, portanto, contribuir para a redução do número de defeitos com relação ao tratamento de exceção - uma das principais causas de falhas de software.

Estender a ferramenta SAFE para outros idiomas orientados a aspectos – Atualmente, existe uma versão da SAFE para *AspectJ* e outra para

JBoss AOP. Um trabalho interessante para o futuro é ampliar o número de linguagens que a ferramenta SAFE suporta, a fim de possibilitar a análise de outras linguagens orientadas a aspectos baseadas em *Java*, como: *CaesarJ, Spring AOP*. Uma maneira de fazer isso é a integração com *CAPE* (*Common Aspect Prove Enviroment*) (Faitelson e Katz, 2008) uma plataforma extensível para a integração de ferramentas de análise e verificação de programas orientados a aspectos. O ambiente de prova *CAPE* permite uma ferramenta concebida para analisar programas orientados a aspectos a executar em uma variedade de outras línguagens.

Detalhar o tratamento de exceção – A ferramenta SAFE permite a classificação dos fluxos como não-capturada, capturada, encapsulada, substituída e relaçada. Contudo, identificando quais as ações que o código de tratamento de exceção realiza, nos forneceria informações detalhadas a respeito do funcionamento do mecanismo de tratamento de exceções das aplicações.