Pontifícia Universidade Católica
do Rio de Janeiro

Roberta de Souza Coelho

Analyzing Exception Flows of Aspect-Oriented Programs

PhD Thesis

Dissertation presented to Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática as part of the requirements for the fulfillment of the doctor's degree in Computer Science.

Supervisor: Arndt von Staa
Co-supervisor: Awais Rashid

Rio de Janeiro
June, 2008

Pontifícia Universidade Católica
do Rio de Janeiro

Roberta de Souza Coelho

Analyzing Exception Flows of Aspect-Oriented Programs

Thesis presented in the Graduate Program in Computer Science of the Pontifícia Universidade Católica do Rio de Janeiro in partial fulfillment of the requirements for the degree of Doctor in Computer Science. Approved by the following Examination Committee.

Prof. Arndt Von Staa
Orientador
Departamento de Informática – PUC-Rio

Prof. José Carlos Pereira de Lucena
Departamento de Informática – PUC-Rio

Prof. Renato Fontoura de Gusmão Cerqueira
Departamento de Artes e Design – PUC-Rio

Prof. Paulo Henrique Monteiro Borba
Centro de Informática – UFPE

Prof. Paulo Cesar Masiero
Instituto de Ciências Matemáticas e de Computação – USP

Prof. José Eugenio Leal
Graduate Programs´ Coordinator for the Center of Science and Techonology – PUC-Rio

Rio de Janeiro, June 11th, 2008

Roberta de Souza Coelho

Graduou-se no Curso de Bacharelado em Ciência da Computação da Universidade Federal de Campina Grande (UFCG) em 1999. Obteve o título de Mestre em Ciência da Computação na Universidade Federal de Pernambuco (UFPE) em 2002. Atuou como coordenadora de projetos, engenheira de software no Centro de Estudos e Sistemas Avançados do Recife (CESAR) de 1999 a 2003. Ainda em 2003 atuou como pesquisadora no projeto Ourgrid/UFCG-HP. Atuou como pesquisadora na área de Desenvolvimento de Software Orientado a Aspectos e Teste de Software no Laboratório de Engenharia de Software (LES/PUC-Rio) de 2004 a 2007, e no Tecgraf/PUC-Rio em 2006.

Ficha Catalográfica

Coelho, Roberta de Souza

Analyzing exception flows of aspect-oriented programs / Roberta de Souza Coelho ; supervisor: Arndt Von Staa; co-supervisor: Awais Rashid. – 2008.

173 f. ; 30 cm

Tese (Doutorado em Informática)–Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2008.

Inclui bibliografia

1. Informática – Teses. 2. Tratamento de exceções. 3. Programas orientados a aspectos. 4. Análise estática. 5. Estudo empírico. 6. Exceções não capturadas. 7. Padrões de erro. I. I. Staa, Arndt von. II. Raschid, Awais. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

A Deus por sempre estar presente em minha vida
através do amor da minha família e amigos.

To God for always being present in my life
through the love of my family and friends.

Acknowledgment

First, I give thanks to God for giving me the greatest gift of my life: my beloved family. To my dear parents Fatima and Roberto for loving and supporting me since I was born. Their love has always played a fundamental role on the achievements of my life. Even living several kilometers apart during my PhD, they were always very present! To my husband Uirá for his love and for supporting me along the PhD journey. God helped us to overcome the difficulties inherent to PhDs – he was also doing his PhD at the same time ;-). To my beloved sister Renata (Nini) and dearest brother Jaime (Jaimão) for their love, care and words of wisdom. I love them so much!

To my advisor, Arndt von Staa who was always very supportive, and whose "investigative spirit" was a great inspiration to this PhD work. Our discussions about the real pros and cons of Aspects were the seeds of this thesis. Besides inspiring my PhD work, this same spirit also inspired other great empirical studies carried out by the Software Engineering Laboratory (LES).

To Professor Awais Rashid for the great discussions on AO development and for advising me during the seven-month exchange period at Lancaster University.

To Professor Carlos Lucena for his support and for creating opportunities of research collaboration.

To my friend and PhD colleague Fabiano Ferrari who contributed on the empirical study performed in this work – helping me to manually inspect the exception handling code of several system versions. To my friend and PhD colleague Cláudio (Baiano) who also contributed with great discussions. To my friend Elder Cirilo who implemented one of the tools developed during the PhD, and to the LES research group for the discussions over those years (on Friday's Seminars). To my dear husband and PhD colleague Uirá for our interesting technical discussions about aspects and testing.

To my dearest grandmother, Lilia, for always loving and motivating me. She is always very lively and funny, and often called me to say: "Dear Robertinha, you should finish your studies and come back home. I miss you. You have studied too much already ;-)!". She was always a great motivation to me.

To Maité (Teté) and Nandinho for opening not only the doors of their home but also to the doors of their hearts, and enabling us to live great moments together with them and my dear nephew Cauê. To D. Tereza e S. Tek for their care and attention.

To my friends from Centro Loyola de Fé e Cultura and Centro de Pastoral

Anchieta da PUC-Rio, especially to: my dearest friend Fernanda Motta (Fê), who shared with me words of wisdom and her friendship; Helena, for being so peaceful and friendly; and Pe Emmanuel, a blessed friend, who is a real image of Christ in our days.

To all my friends from Chaplaincy Center at Lancaster University specially for Pr. Hugh, Sister Ella, Jim, Rosa, Veronica, Robert, Kersting, Gosia and Mariel. They represented real gifts from God while I was in Lancaster (The Friendship Icon)

To my friends from Tecgraf: Julia, Leticia, Clarissa, Taciana, Malu, Cris, Ana Moura, Ana and Andrea. We formed a software engineering group composed mostly by women – which is very rare is computer science. It was a very special team! Really friendly and competent! My experience at Tecgraf was was one of the inspirations of my research on exception handling.

To my friends Silvia Passos, Mariza Lucena and Vera Menezes for their friendship and sympathy!

To my friends from the Northeast, who also came to Rio study at PUC, Silvinha, Suzana. We laugh a lot, and shared great moments and the same accent!

To my friends from DobrasBrasil: Sol (*in memorium*), a Amália, Jaciara e Glória. Their enthusiasm and happiness made the Saturday's afternoons very special.

To AOSD Research group of Lancaster University especially Alessandro and Nélio for their feedback on the work presented here and for the interesting discussions concerning exception handling.

To the professors of the committee, Paulo Borba, Renato Cerqueira e Paulo Masiero for giving great suggestions on how to improve this thesis.

To the Informatics Department, its professors and the technical staff, for supporting my research, and to CNPq, CAPES and FAPERJ for funding it.

Abstract

Coelho, Roberta; von Staa, Arndt. **Analyzing Exception Flows of Aspect-Oriented Programs.** Rio de Janeiro, 2008. 173p. PhD Thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

It has been empirically observed that aspect-oriented (AO) decompositions promote the modularity and the design stability of software systems containing crosscutting concerns. However, most of this existing empirical research has focused on the positive and negative impacts of AO programming in the modularization of crosscutting concerns in the context of "normal" control flow of programs. Consequently, most of these works do not account for the exceptions that may flow from aspects: when an aspect adds a new functionality to specific points in the code, this additional behavior may also bring new exceptions. An aspect also has the ability to handle exceptions that were previously handled inside the base code. Moreover, the exceptions that escape form aspects also flow inside the program, and may lead to unexpected error-prone scenarios, such as: unintended handler actions and uncaught exceptions. This thesis presents an empirical study to evaluate the impact of aspects on the exception flow of programs. To support the reasoning about the flow of exceptions on AO programs, a static analysis tool called SAFE (Static Analysis for the Flow of Exceptions) was implemented. Based on data empirically collected during the study we characterized a catalogue of *bug patterns* related to the exception handling code of AO programs. To help AO developers to check the reliability of the exception handling code, this work presents verification approach - supported by the SAFE tool - which provides guidelines to counter these *bug patterns*. Our findings show that the exception handling code in aspect-oriented programs tends to be error-prone, but that a verification approach based on static analysis can lead to significant improvements.

Keywords

Exception handling, aspect-oriented programs, static analysis, empirical study, uncaught exceptions, bug patterns.

Resumo

Coelho, Roberta; von Staa, Arndt. **Analisando o Fluxo de Exceções em Programas orientados a Aspectos.** Rio de Janeiro, 2008. 173p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Os mecanismos de tratamento de exceções têm o objetivo de aumentar a robustez e a modularidade do software na medida em que promovem a separação entre o código dedicado ao tratamento de erros e código associado ao fluxo normal de execução do programa. Estes permitem a detecção de erros e a associação de respostas adequadas aos mesmos, através da execução de códigos de recuperação que são geralmente encapsulados em tratadores (do inglês: *handlers*). A importância dos mecanismos de tratamento de exceção é atestada pelo fato destes fazerem parte da maioria das linguagens de programação mais utilizadas na atualidade: Java, C++ e C#.

Na última década a programação orientada a aspectos (POA) vem sendo amplamente utilizada como forma de modularizar interesses que se encontram espalhados nas decomposições primarias de um sistema (e.g., funções, classes, métodos) em uma abstração chamada *aspecto*. Os aspectos utilizam construções especificas para promover modificações externas nos programas, incluindo comportamentos adicionais em pontos específicos do código. POA vem sido utilizado com o objetivo de aumentar a modularidade do código de tratamento de exceção e de interesses transversais igualmente importantes tais como: persistência, distribuição, segurança, controle de transações e monitoramento.

De acordo com alguns estudos empíricos, POA tem sido usado com sucesso com o objetivo de promover o tratamento de erro modular em vários cenários de tratamento de exceções. Porém, é sabido que mecanismos de programação flexíveis (e.g., herança e polimorfismo) podem ter um efeito negativo no tratamento de exceções. Se por um lado os mecanismos de composição POA baseados na inversão de controle podem trazer um novo leque de possibilidades de projeto, promovendo em muitas circunstâncias uma maior estabilidade do projeto (do inglês: *design stability*), eles podem perder seu valor se ele torna o tratamento de exceções propenso a erros (do inglês: *error-prone*). Refinamento aspectuais do comportamento do código base, podem por um lado promover a robustez do sistema em situações onde exceções são lançadas ou contribuir para os problemas típicos de tratamento de exceções mal projetados tais como: (i) exceções não capturadas (do inglês: *uncaught exceptions*); (ii) exceções

capturadas por tratadores genéricos (do inglês: *exception subsumption*) (i.e., exceções capturadas por um tratador cujo argumento é um supertipo da exceção a ser tratada); (iii) e exceções capturadas por tratadores errados (do inglês: *unintended handler action*).

Infelizmente, não existe nenhuma avaliação sistemática dos efeitos positivos e negativos da programação orientada a aspectos na robustez do código de tratamento de exceções. Os trabalhos de pesquisa descritos na literatura têm se limitado a analisar os impactos dos aspectos no fluxo normal de execução do programa. Além disso, a maioria dos estudos empíricos não vai além de discussões sobre os ganhos de modularidade e as ciladas associadas à utilização de aspectos - para o tratamento de exceções e de outros interesses transversais. Por exemplo, estes estudos não levam em consideração as conseqüências inerentes as novas exceções e tratadores que são adicionadas ao código base junto com a nova funcionalidade adicionada pelos aspectos.

Esta tese descreve o primeiro estudo sistemático que avalia quantitativamente o impacto da composição aspectual no fluxo excepcional de programas. Este estudo se baseou na utilização de uma ferramenta de análise do fluxo de exceções chamada SAFE (do inglês: *Static Analysis for the Flow of Exceptions*) desenvolvida ao longo deste trabalho e na inspeção manual do código de tratamento de exceções de um conjunto de sistemas para os quais a versão Java e AspectJ estavam disponíveis. A ferramenta SAFE foi construída com base no framework SOOT para a análise e otimização de *bytecode*. A partir de uma representação do programa construída a partir do *bytecode* do código base combinado com os aspectos (do inglês: *woven bytecode*), a ferramenta calcula estaticamente os caminhos excepcionais do programa (i.e., caminhos no grafo de chamadas que ligam o elemento que lança uma exceção ao elemento que contém o tratador que irá capturá-la). Estes caminhos são então armazenados servindo como base para as atividades de inspeção manual do código de tratamento de exceções.

Três sistemas de médio porte foram avaliados neste estudo: Health Watcher – um sistema Web que permite o registro de queixas para sistema público de saúde; o Mobile Photo – uma linha de produto de software para aplicações de manipulação de media (arquivos de som, imagem e vídeo) em dispositivos móveis; e o JHotDraw – um framework *open source* para construção de aplicações de edição gráfica. Para os dois primeiros sistemas mais de *release* foi avaliado. Ao todo 10 versões foram avaliadas, totalizando 41.1 KLOC de código Java dos quais aproximadamente 4.1 KLOC eram dedicados ao

tratamento de exceções, e 39 KLOC linhas de código AspectJ dos quais aproximadamente 3.2 KLOC eram dedicadas ao tratamento de exceções.

Alem de representarem aplicações de diferentes domínios e as aplicações analisadas neste estudo possuem estratégias de tratamento de exceções diferentes. No Health Watcher apenas o código de tratamento de exceções das funcionalidades transversais (representadas por aspectos) foram movidas para aspectos de tratamento de exceções (*Exception Handling Aspects*). No Mobile Photo a maioria do código de tratamento de exceções foi *aspectualizado* – não apenas o tratamento de exceções dos interesses transversais. Já no JHotDraw, o código de tratamento de exceções da versão AspectJ permaneceu no código base.

Alguns resultados negativos foram detectados de forma consistente através da análise das *versões* AO de cada sistema, quais sejam: (i) um maior número de caminhos excepcionais que continham exceções não capturadas (do inglês: *uncaught exceptions*) foi encontrado nas versões que usavam aspectos como tratadores de exceções - exceções não capturadas podem levar a *crashs* no programa de forma imprevisível; e (i) um maior número de exceções capturadas por *subsumption* algumas delas levando ao tratamento inadequado de exceções, i.e, exceções lançadas por aspectos e capturadas erroneamente por tratadores presentes no código base.

Neste trabalho investigamos as causas para tais aumentos e apresentamos as mesmas na forma de um catálogo de padrões de erro (do inglês: *bug pattern*) relacionado ao código de tratamento de exceções de programas AspectJ. Além disso, este trabalho propõe uma abordagem de verificação, baseada na utilização da ferramenta SAFE, com o objetivo de auxiliar os desenvolvedores na detecção e solução de potenciais faltas no código de tratamento de exceções de sistemas AO.

São, portanto, contribuições deste trabalho: (i) a realização da primeira análise sistemática com o objetivo de investigar como os aspectos afetam o fluxo excepcional dos programas; (ii) a construção de uma ferramenta de análise estática do fluxo excepcional de sistemas Java e AspectJ; (iii) a definição de um catálogo padrões de erro associado ao código de tratamento de exceções de sistemas AspectJ, o qual foi obtido através de dados empiricamente coletados; e (iv) a definição e validação inicial de uma abordagem de verificação, baseada na utilização da ferramenta SAFE para detecção e solução de potenciais faltas

no código de tratamento de exceções de sistemas AO.

Os resultados deste trabalho mostraram que o código de tratamento de exceções de programas orientados a aspectos é, de fato, propenso a falhas; mas uma abordagem de verificação baseada em analise estática pode reduzir significativamente o grau em que estas falhas acontecem. As contribuições alcançadas neste trabalho permitirão que: (i) desenvolvedores de aplicações AO avaliem o impacto causado pelos aspectos no fluxo excepcional do programa ao refatorar ou implementar um sistema AO; e (ii) projetistas de linguagens AO considerem a realização de refinamentos nos mecanismos de tratamento de exceções existentes, de forma a torná-los mais robustos e resilientes aos tipos de erros encontrados neste estudo.

Palavras-chave

Tratamento de exceções, programas orientados a aspectos, análise estática, estudo empírico, exceções não capturadas, padrões de erro.

Index

List of Figures

List of Tables

List of Abbreviations

AOSD	Aspect Oriented Software Development
AOP	Aspect Oriented Programming
AO	Aspect Oriented
OO	Object Oriented
GUI	Graphical User Interface
XPI	Crosscutting Interface
HW	Health Watcher system
MP	Mobile Photo system
JHD	JHotDraw system