



Márcio David de Magalhães Santos

Estendendo a Ferramenta SAFE para JBOSS AOP

Dissertação de Mestrado

Dissertação apresentada ao Programa de Pós-Graduação em Informática da PUC-Rio como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Prof. Arndt von Staa
Co-Orientadora: Prof^a. Roberta Coelho

Rio de Janeiro
Agosto de 2010



Márcio David de Magalhães Santos

Estendendo a Ferramenta SAFE para JBOSS AOP

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Arndt von Staa

Orientador

Departamento de Informática – PUC–Rio

Prof. Renato Fontoura de Gusmão Cerqueira

Departamento de Informática – PUC–Rio

Profª Simone Diniz Junqueira Barbosa

Departamento de Informática – PUC–Rio

Prof. José Eugenio Leal

Coordenador Setorial do Centro

Técnico Científico – PUC-Rio

Rio de Janeiro, 20 de Agosto de 2010

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

Márcio David de Magalhães Santos

Graduou-se em Ciência da Computação pela Universidade Federal de Pernambuco – UFPE, em novembro de 1999. Área de interesse acadêmico: Engenharia de Software.

Ficha Catalográfica

Santos, Márcio David de Magalhães

Estendendo a Ferramenta SAFE para JBOSS AOP / Márcio David de Magalhães Santos; orientador: Arndt von Staa – 2010

147 f ; 30 cm

Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2010.

Inclui bibliografia.

1. Informática – Teses. 2. Engenharia de Software. 3. Tratamento de Exceções. 4. Programas Orientados a Aspectos. 5. Análise Estática. 6. Estudo Empírico. 7. Exceções não Capturadas. 8. JBoss AOP I. von Staa, Arndt. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Dedico esta dissertação aos meus orientadores Arndt e Roberta pela dedicação e paciência. Aos meus pais pelo apoio e incentivo, e aos amigos que tanto ajudaram durante o mestrado.

Agradecimentos

Agradeço ao Uirá pelo incentivo para fazer esta dissertação; a Elizabeth, Katia e Paola pelo tempo que estudamos juntos; a Hegmann, Rita e Marcelo pelo suporte

Resumo

Santos, Márcio David de Magalhães; von Staa, Arndt. **Estendendo a Ferramenta SAFE para JBOSS AOP**. Rio de Janeiro, 2010. 147p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O paradigma de orientação a aspectos é utilizado em vários frameworks e aplicações com o objetivo de melhorar a modularidade e a separação de interesses. Contudo, a combinação do paradigma de orientação a aspectos com mecanismos de captura de exceção pode elevar o número de cenários sujeitos a falhas. Pois, os elementos que compõem os aspectos podem levantar exceções, as quais a aplicação não foi projetada para tratá-las. Nesta dissertação é apresentado (i) um estudo empírico mostrando como a programação orientada a aspectos afeta o tratamento de exceção de uma aplicação que utiliza o JBoss AOP como implementação do paradigma de orientação a aspectos; e (ii) uma ferramenta que deu suporte ao estudo. O estudo mostra que ocorrem falhas no tratamento de exceção principalmente porque as exceções são capturadas por subsunção.

Palavras-chave

Tratamento de Exceções; Programas Orientados a Aspectos; Análise Estática; Estudo Empírico; Exceções não Capturadas; JBoss AOP

Abstract

Santos, Márcio David de Magalhães; von Staa, Arndt. (Advisor). **Extending the Tool SAFE for JBOSS AOP**. Rio de Janeiro, 2010. 147p. MSc. Dissertation – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Aspect Oriented Paradigm (AOP) is used in many frameworks and applications in order to improve the modularity and separation of concerns. However, the combination of AOP and exception handling mechanisms may increase the number of error-prone scenarios. AOP may raise exceptions which the application was not designed to handle it. This dissertation presents (i) an empirical study showing how the AOP affects exception handling in an application that uses JBoss AOP to implement AOP, and (ii) a support tool for the study. The study shows that error-prone scenarios occur mainly because exception handling exceptions are caught by subsumption.

Keywords

Exception Handling; Aspect-oriented Programs; Static Analysis; Empirical Study; Uncaught Exceptions; JBoss AOP

Sumário

1 Introdução	15
1.1. O Problema	16
1.2. Objetivos	17
1.3. Estrutura do Documento	18
2 Conceitos Básicos	19
2.1. JBoss AOP	19
2.1.1. Aspecto	21
2.1.2. Pointcut	25
2.1.3. Advices	27
2.1.4. Binding	29
2.1.5. Introductions / Mixing	30
2.1.6. Weaving	30
2.2. Mecanismos de Tratamento de Exceções	36
2.2.1. Mecanismos de Tratamento de Exceção em Programas JBoss AOP	37
2.2.2. Construções para Captura de Exceções em JBoss AOP	41
2.3. JBoss AS	44
2.3.1. EJB	46
2.4. SAFE	48
2.5. Resumo	49
3 Ferramenta SAFE-JBoss AOP	51
3.1. Técnicas Empregadas	52
3.1.1. Forward Analysis	52
3.1.5. Identificando Advices	59
3.1.6. Point-to	62
3.2. Ferramenta SAFE-JBossAOP	67
3.2.1. Arquitetura da Ferramenta	67

3.2.3. Funcionamento do Algoritmo na Presença de Aspectos do JBoss AOP	81
3.2.4. Saída da Ferramenta	86
3.2.5. Testes	90
3.3. Resumo	91
4 Caracterizando o Fluxo de Exceções em Aplicações JBoss-AOP: um Estudo Exploratório	93
4.1. Caracterizando o Estudo	93
4.1.1. Aplicações Estudadas	94
4.2. Execução do Estudo	99
4.3. Dados Obtidos e Interpretação dos Mesmos	100
4.3.1. Dados Extraídos da Ferramenta SAFE-JBossAOP	101
4.4. Padrões de Tratamento de Exceção Encontrados	108
4.4.1. Encapsulamento de Exceções em RuntimeException	108
4.4.2. Captura de java.lang.Exception na Camada de Apresentação	110
4.5. Restrições do Estudo	111
4.6. Resumo	111
5 Trabalhos Relacionados	113
5.1. Ferramentas para análise estática	113
5.2. Estudos Empíricos	115
5.3. Resumo	117
6 Conclusão e Trabalhos Futuros	118
6.1. Conclusões	118
6.2. Contribuições	120
6.3. Trabalhos Futuros	120
7 Referências Bibliográficas	122
8 Anexos	128
8.1. Exemplo da Saída da Ferramenta	128
8.2. Lista das Exceções da Aplicação CMS	132

8.3. Lista das Exceções da Aplicação WordNet	135
8.4. Lista das Exceções da Aplicação Health Care	140
8.5. Classe Simples Após o Weaving	142
8.6. Código modificado pela instrumentação Generated Advisor Weaving	144

Lista de figuras

Figura 1 – Exemplo de código de um aspecto utilizando JBoss AOP.	21
Figura 2 – Exemplo de definição de aspectos usando XML.	22
Figura 3 – Implementação de uma fábrica de aspectos.	23
Figura 4 – Exemplo da configuração de domínios no JBoss AOP.	24
Figura 5– Exemplos de uma aplicação afetada por dois domínios diferentes.	25
Figura 6 – Exemplo de advice recebendo o contexto de execução.	29
Figura 7 – Exemplo de adição de interfaces a classes interceptadas.	30
Figura 8 – Exemplo de uma classe simples.	32
Figura 9 – Exemplo de um advice.	32
Figura 10 – Exemplo da configuração do JBoss AOP para que o aspecto da Figura 7 intercepte os métodos da classe da Figura 6.	32
Figura 11 – Diagrama de classes da hierarquia de Invocation.	34
Figura 12 – Mapeamento entre o valor original no arquivo o valor que o substituiu.	36
Figura 13 – Exemplo de fluxo de exceção na presença de aspectos.	39
Figura 14 – Hierarquia de exceções na linguagem Java.	42
Figura 15 – Trecho do descritor de implantação da aplicação health care.	47
Figura 16 – Representação de dois fluxos de chamadas independentes entre si.	53
Figura 17 – Representação de dois fluxos de chamadas independentes entre si e em presença de um aspecto que intercepta os dois fluxos.	54
Figura 18 – Representação do fluxo de exceção a ser analisado em presença de aspectos.	55
Figura 19 – Exemplo para ilustrar o funcionamento do bloco finally.	56
Figura 20 – Código descompilado com a ferramenta Jad.	56
Figura 21 – Exemplo do encapsulamento de uma exceção.	57
Figura 22 – Exemplo da passagem de dados de uma exceção para outra.	57
Figura 23 – Função para cálculo do número de Fibonacci.	58
Figura 24 – Função para cálculo de métricas.	58
Figura 25 – Execução do algoritmo de Fibonacci para o valor 5.	59
Figura 26 – Trecho de código que mostra o código descompilado de um método que foi interceptado por um aspecto.	61
Figura 27 – Arquitetura da ferramenta.	68
Figura 28 – Trecho do arquivo ejb3-interceptors-aop.xml.	70
Figura 29 – Exemplo do arquivo de configuração ejb-jar.xml da aplicação hospital care.	73
Figura 30 – Pseudocódigo do ponto de inicialização do algoritmo que analisa um método.	75

Figura 31 – Pseudocódigo do método “analisa” do algoritmo que analisa um método.	76
Figura 32 – Pseudocódigo do método “métodos reais” do algoritmo que analisa um método.	77
Figura 33 – Exemplo de um método sobrescrito pela subclasse.	79
Figura 34 – Trecho de código para exemplificar as distorções causadas pelo algoritmo CHA.	80
Figura 35 – Exemplo de captura de exceção.	83
Figura 36 – Exemplo da listagem se saída da ferramenta.	84
Figura 37 – Exemplo de relançamento de exceção.	85
Figura 39 – Exemplo de encapsulamento de uma exceção por adição de atributo da exceção original.	86
Figura 40 – Exemplo da substituição da exceção por outra.	86
Figura 41 – Exemplo de impressão de fluxos de exceção da ferramenta.	90
Figura 42 – Representação arquitetural do aplicativo CMS.	95
Figura 43 – Representação arquitetural da aplicação WordNet.	97
Figura 44 – Representação arquitetural da aplicação COS-HMS.	98
Figura 45 – Número de fluxos de exceção por sistema.	103
Figura 46 – Número de exceções lançadas por profundidade.	104
Figura 47 – Tratamento de exceção de exceções originadas em aspectos.	106
Figura 48 – Exemplo de encapsulamento de exceções checadas em exceção não-checada na classe org.jboss.ejb3.EJBContainer.	109
Figura 49 – Exemplo de encapsulamento da exceção Throwable em exceção não-checada na classe org.jboss.ejb3.EJBContainer.	110

Lista de tabelas

Tabela 1 – Resumo Funções da Linguagem.	26
Tabela 2 – Anotações utilizadas para associar valores da execução a parâmetros no método do advice.	28
Tabela 3 – Representação da heap quando um valor é atribuído a um atributo.	63
Tabela 4 – Representação da heap quando um valor é atribuído a um atributo.	64
Tabela 5 – Representação da heap para exemplificar associação simples da aplicação A1.	64
Tabela 6 – Representação da heap para exemplificar associação com cast da aplicação A1.	65
Tabela 7 – Representação da heap quanto a análise inter-procedural.	65
Tabela 8 – Representação da heap quando uma variável recebe o retorno de um método.	66
Tabela 9 – Aspectos definidos para os domínios statelessSessionBean e StateFulSessionBean.	71
Tabela 10 – Aspectos definidos para o domínio MessageDrivenBean.	71
Tabela 11 – Mapeamento de tipo de bean para domínio de aspectos.	72
Tabela 12 – Estrutura das estatísticas em memória.	78
Tabela 13 – Tabela para ordenação da chamada de métodos e aspectos.	82
Tabela 14 – Características do código das aplicações analisadas.	99
Tabela 15 – Listagem dos filtros utilizados na ferramenta por aplicativo.	100
Tabela 16 – Classificação dos fluxos de exceção por aplicação.	102
Tabela 17 – Número de fluxos de exceção por aspecto.	107
Tabela 18 – Interesses transversais das aplicações analisadas por Coelho (2008).	117

Lista de Abreviações

CMS	Conference Management System
COS-HMS	COS Health Management System
EJB	Enterprise Java Beans
GUI	Graphical User Interface
OA	Orientado a Aspectos
OO	Orientado a Objetos
POA	Programação Orientada a Aspectos
WN	WordNet