4
Caracterizando o Fluxo de Exceções em Aplicações JBoss-AOP: um Estudo Exploratório

Esta seção tem por finalidade descrever o presente estudo em termos de seus objetivos e hipóteses, além de tratar, também, sobre os critérios utilizados para a seleção de sistemas de destino (Seção 4.1), a metodologia empregada para realizar as análises de código excepcional (Seção 4.2) e a execução efetiva do nosso estudo (Seção 4.3). Assim, temos por objetivo avaliar o impacto do JBoss AOP nos fluxos de programas que o utilizam e essa investigação baseia-se em determinar os fluxos de exceção que são tipicamente introduzidos na aplicação pelo JBoss AOP.

As subseções seguintes, por sua vez, detalham como este estudo foi elaborado, quais as aplicações utilizadas, quais foram os procedimentos de excecução da análise, quais os dados extraídos e qual a interpretação dos mesmos e, por fim, quais as restrições aplicadas no estudo.

4.1.Caracterizando o Estudo

Neste estudo de caso, foram analisadas três aplicações e suas respectivas bibliotecas, a fim de observar os efeitos causados pelos aspectos JBoss AOP, e adicionados pelo framework EJB, têm sobre seus fluxos de exceção. Na presente dissertação, foram adotados alguns procedimentos para distinguir fluxos originados a partir de aspectos dos AOP dos que são gerados a partir de classes e bibliotecas utilizadas pela aplicação, além de diferenciar, também, os fluxos entre os aspectos, de forma a detectar a fonte de exceções das aplicações. Além disso, fazemos uma distinção entre os cenários de exceção que podem ocorrer, tais como (i) exceções não capturadas, (ii) exceções relançadas, (iii) exceção encapsuladas, (iv) exceções substituídas, e (v) exceções capturadas por subsunção, mostrando, desta forma, o funcionamento do fluxo de exceções na aplicação. Esses procedimentos, portanto, foram importantes para detectar que aspectos podem levar a cenários inesperados. Assim, optamos por realizar um estudo empírico ao

PUC-Rio - Certificação Digital Nº 0812598/CA

invés de um estudo controlado, por que num estudo empírico temos pouco controle sobre os sistemas, uma vez que foram desenvolvidos por terceiros, de forma que eles representam situações reais do uso do JBoss AOP.

Sendo assim, a hipótese nula (H0) para o estudo empírico é a de que o framework de aspectos JBossAOP injeta exceções na aplicação que não são corretamente capturadas. Além disso, devido a todos os sistemas analisados utilizarem o framework EJB, adicionamos a hipótese (H1) ao estudo empírico de que o framework EJB adiciona aspectos desnecessários às aplicações que o utiliza, de forma a afetar o tratamento de exceção.

4.1.1. Aplicações Estudadas

A seleção das aplicações que participam deste estudo é importante para a qualidade da análise. Encontrar aplicações que possuam código disponível na Internet e que utilizem o JBoss AOP como o framework de aspectos de forma isolada é raro, mas o servidor de aplicações JBoss AS utiliza o JBoss AOP para implementar os interesses transversais do framework EJB que, por sua vez, é utilizado por muitos sistemas corporativos.

As aplicações selecionadas foram três sistemas de pequeno porte (CMS, WordNet (WN) e COS-HMS (COS)), mas que representam as arquiteturas utilizadas pela maioria dos sistemas corporativos, além de ser, cada um deles, um representante de diferentes domínios de aplicação. O sistema de CMS (Conference Management System) é um aplicativo Web utilizado para gerenciar conferências e publicações relativas a elas. Já o WordNet é um produto que disponibiliza uma interface Web com a finalidade de gerenciar um banco de dados de palavras e os relacionamentos entre elas. E, por fim, o COS é um sistema que possui uma interface Web que permite o gerenciamento hospitalar. Vale ressaltar, ainda, que todas as aplicações acima são de código aberto disponíveis na Internet.

4.1.1.1.CMS

O Conference Management System (CMS) foi desenvolvido pela universidade de Drexel e permite o gerenciamento do ciclo de vida de uma conferência que engloba desde o envio dos artigos até o gerenciamento das revisões efetuadas e a eventual publicação do artigo na conferência.

PUC-Rio - Certificação Digital Nº 0812598/CA

O CMS é uma aplicação Web que utiliza o padrão arquitetural camada (BUCHMAN et alii, 1996) para separar os interesses relativos à interface gráfica e à persistência. A primeira é implementada com os frameworks Struts2[6] e XWork[7] e os aspectos que interceptam a aplicação são provenientes do framework EJB e interceptam a classe `CmsDbQuery` adicionando ao comportamento da classe interesses como (i) o gerenciamento de transações, (ii) as chamadas assíncronas e (iii) a segurança.

A Figura 42 apresenta uma visão arquitetural da aplicação CMS. A classe `ReleaseReviewAction` representa as ações que são realizadas pelo usuário e acessa a classe `CmsDao` que funciona como um adaptador do padrão Adapter (GOF, 1995) para acesso ao mecanismo de persistência da aplicação que é realizado pela classe `CmsDbBean`.

PUC-Rio - Certificação Digital Nº 0812598/CA

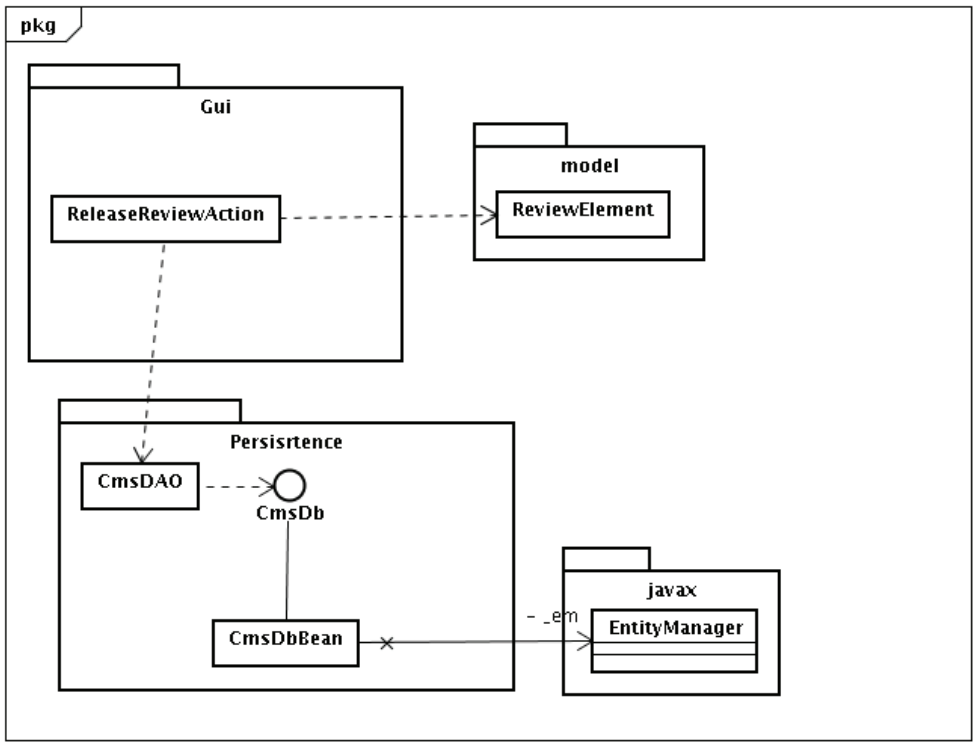

Figura 42 – Representação arquitetural do aplicativo CMS.

[6] Struts2 é um framework para gerenciamento de requisições do usuário em sistemas web. http://struts.apache.org/2.x/

[7] Xwork é um framework usado adicionar a capacidade de inversão de controle no framework Struts2.

4.1.1.2.WordNet

O WordNet® (WN) é uma aplicação que gerencia uma base de dados lexicais do Inglês e que é utilizada como ferramenta para a linguística computacional e processamento de linguagem natural desenvolvido na universidade de Princeton[8], sendo uma aplicação Web assim como a CMS.

O WN foi desenvolvido em três camadas (BUCHMAN et alii, 1996), para separar os interesses relativos à interface gráfica, às regras de negócio e à persistência. A interface gráfica é implementada com o framework Java Servlet que, por sua vez, acessam a interface `WordNet` que funciona de forma semelhante ao padrão Facade (GOF, 1995) e é a partir desta que são acessadas as classes de persistência. A fachada da aplicação é implementada utilizando o framework EJB, responsável pelo gerenciamento de transações da aplicação.

A Figura 43 mostra uma visão arquitetural da aplicação WordNet em que o pacote `web` representa as ações que são realizadas pelo usuário e acessa a fachada (GOF, 1995) que contém a regra de negócio da aplicação. E esta utiliza classes que implementam o pattern Command (GOF, 1995) para acessar o banco de dados, de forma que estas classes sejam responsáveis por realizar o mapeamento objeto-relacional e representar as entidades persistentes da aplicação.

PUC-Rio - Certificação Digital Nº 0812598/CA

[8] Mais informações a respeito do WordNet podem ser encontradas em http://wordnet.princeton.edu/

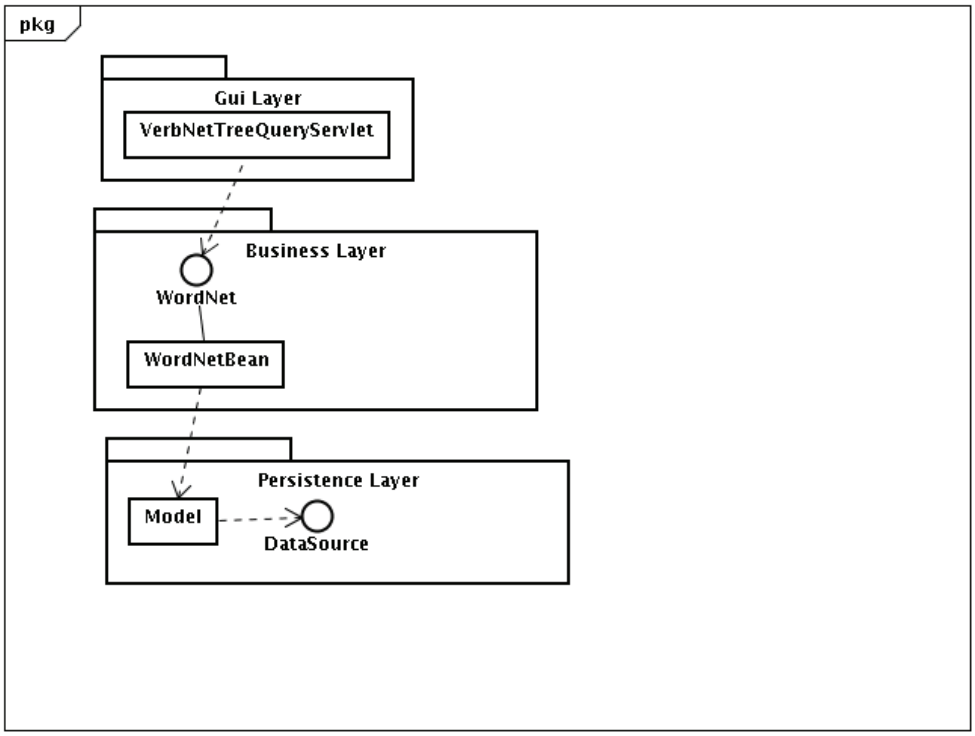

Figura 43 – Representação arquitetural da aplicação WordNet.

4.1.1.3.COS-HMS

A aplicação COS-HMS é um sistema de gerenciamento hospitalar, que controla o estoque de medicamentos, entrada e saída de pacientes, a execução de exames laboratoriais e radiológicos, além de prover relatórios gerenciais.

Com relação à arquitetura da aplicação, a mesma foi desenvolvida para a Web e, assim como as demais aplicações, utiliza o padrão arquitetural camada (BUCHMAN et alii, 1996) para separar os interesses relativos à interface gráfica e persistência. Uma particularidade desta aplicação é que a regra de negócio está implementada no banco de dados tornando, desta forma, a aplicação dependente do banco de dados MySql.[9] Neste caso, a interface gráfica é implementada com o framework Java Servlet e sua aplicação é separada em módulos: (i) `administration`, módulo responsável por gerenciar o cadastro dos usuários da aplicação; (ii) `radiology`, módulo responsável por gerenciar os exames radiológicos da unidade; (iii) `patology`, que tem por responsabilidade gerenciar os exames clínicos; (iv) *`emergency`*, responsável por gerenciar os medicamentos

PUC-Rio - Certificação Digital Nº 0812598/CA

e a entrada e saída de pacientes do hospital; e (v) `domain`, módulo responsável pelo cadastro das entidades de domínio utilizadas pela aplicação. Além disso, cada módulo da aplicação possui seus respectivos Servlets que gerenciam a navegação do usuário pela aplicação e invocam as classes relativas aos serviços do respectivo módulo acessando, assim, o banco de dados. Portanto, cada serviço da aplicação é implementado com o framework EJB que é, principalmente, utilizado para o gerenciamento de transações da aplicação.

Na Figura 44 é apresentada uma visão do módulo `patology` para exemplificar a arquitetura da aplicação. Sendo assim, a classe *`PthCriticalTestSchAddServlet`* é responsável por receber a requisição do usuário e solicitar à *`ServiceLocator`* a referência para a classe de serviço do módulo para, assim, pedir o processamento dos dados enviados pelo usuário da aplicação. Por fim, a classe de serviço do módulo acessa a interface do banco de dados para persistência ou recuperação de dados.

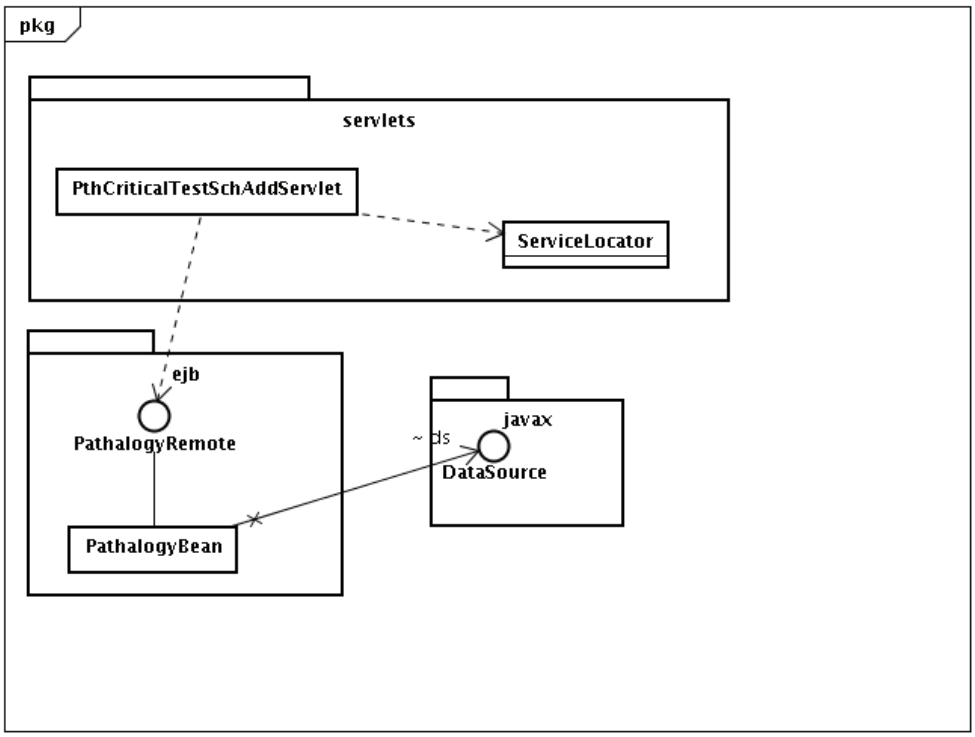

Figura 44 – Representação arquitetural da aplicação COS-HMS.

[9] Mysql é um banco de dados relacional fornecido pela Oracle. http://www.mysql.com

PUC-Rio - Certificação Digital Nº 0812598/CA

4.1.1.4.Características dos Sistemas Estudados

De acordo com o que foi apresentado anteriormente as aplicações utilizam o JBoss AOP através do framework EJB que possui um conjunto de interesses transversais padrão (ver Seção 2.3.1) e, assim, as aplicações são interceptadas sempre pelos mesmos aspectos. Estes aspectos endereçam interesses como segurança, gerenciamento de transação, gerenciamento de contexto, chamadas assíncronas, gerenciamento do ciclo de vida das entidades do container (ver Seção 2.3.1).

As aplicações foram selecionadas, primeiramente, devido às suas diferenças na arquitetura de forma a serem uma amostra representativa das utilizadas em aplicações comerciais. Em segundo lugar, foram selecionadas de acordo com as suas funcionalidades e bibliotecas distintas, de forma a apresentar diferentes fontes de exceções. Por fim, o critério de seleção guiou-se pela variação do tratamento de exceção que vai desde o relançamento de exceções à apresentação de mensagens ao usuário. Porém, em todas as aplicações, as exceções por padrão recebem algum tipo de tratamento. Na Tabela 14, são apresentadas as características do código das aplicações estudadas.

Tabela 14 – Características do código das aplicações analisadas.

	CMS	WordNet	COS-HMS
Número de Classes	69	45	121
LOC	3500	5391	7281
Throws (com biblioteca)	6851	6595	1452

4.2. Execução do Estudo

O presente estudo foi realizado entre maio de 2009 e julho de 2010 e, durante este período, os sistemas a serem analisados foram selecionados e a ferramenta de análise estática foi implementada. A ferramenta tem como saída um arquivo que contém as estatísticas dos fluxos de exceção e a lista de métodos que participaram de cada fluxo (ver Seção 3). A partir destes dados, alguns fluxos foram analisados e selecionados manualmente, pois não seria viável percorrer cada fluxo excepcional da aplicação, uma vez que além do código da aplicação, são analisados os códigos de bibliotecas utilizadas pela mesma. A partir desta

PUC-Rio - Certificação Digital Nº 0812598/CA

análise, portanto, foram detectados padrões de fluxos que seriam impossíveis de ocorrer e, desta forma, tornou-se necessário modificar algumas configurações da ferramenta para cada aplicação com o objetivo de remover tais padrões de fluxos tornando a ferramenta mais precisa em relação ao número de fluxos excepcionais existentes na aplicação. Assim, os filtros aplicados no processo de análise estática da ferramenta são listados abaixo e, na Tabela 15, é apresentada a relação entre aplicação e filtros utilizados na mesma. Por nossa conveniência, configuramos a ferramenta para utilizar o algoritmo sensível ao contexto descrito na Seção 3.2.2.2.

1. **java.lang** – as exceções ocorridas dentro deste pacote podem ser lançadas quase em qualquer instrução de código da aplicação;
2. **javax.swing** – as aplicações foram desenvolvidas para a Web, desta forma não há uso dos componentes Swing;
3. **org.jboss.wsf** – framework da JBoss para webservice, e nenhuma das aplicações utiliza webservice;
4. **hibernate** – biblioteca de mapeamento objeto relacional, algumas aplicações utilizam outras não;
5. **Exceção AssertionErro** – exceção que é introduzida na aplicação para indicar defeitos nela, quando o desenvolvedor está executando testes unitários, esta é provida pela linguagem Java e a ocorrência desta exceção pode ser desativada configurando a máquina virtual.

Tabela 15 – Listagem dos filtros utilizados na ferramenta por aplicativo

Aplicativo	Filtros Aplicados
CMS	1,2,3 e 5
WordNet	1,2,3,4 e 5
COS-HMS	1,2,3 e 5

4.3.Dados Obtidos e Interpretação dos Mesmos

Esta seção apresenta os resultados extraídos de cada aplicação analisada e fornecidos pela nossa ferramenta. Serão apresentados os dados, e, em seguida, será apresentada a interpretação dos mesmos.

PUC-Rio - Certificação Digital Nº 0812598/CA

O objetivo de apresentar os dados é permitir o entendimento de como os aspectos do JBoss AOP e as bibliotecas utilizadas pela aplicação afetam a robustez do tratamento de exceção em cada sistema analisado. Nesta análise, portanto, visamos responder às seguintes perguntas: Quem é o responsável por lançar as exceções? Como são tratadas as exceções lançadas por aspectos? As seções seguintes descrevem os dados empíricos analisados neste estudo e fornecer respostas a tais questões.

4.3.1. Dados Extraídos da Ferramenta SAFE-JBossAOP

A Tabela 16 apresenta o número de fluxos de exceção identificados pela nossa ferramenta (Seção 3). Na tabela é apresentada a contagem de caminhos de exceção por sistema, organizada de acordo com a relação entre o tipo do sinalizador e o manipulador da exceção. O elemento responsável pelo lançamento da exceção tanto pode ser uma classe quanto um aspecto. Quando a exceção é lançada por um aspecto, significa dizer que a exceção é lançada por: um Advice ou por trechos de códigos adicionados à classe interceptada.

Caso uma exceção seja sinalizada, ela pode ser capturada de duas formas (ver Seção 2.2): (i) por um manipulador de exceção especializado, o que ocorre quando o tipo da exceção definido no manipulador de exceções é do mesmo tipo que a exceção que foi lançada; ou (ii) por subsunção, o que ocorre quando o tipo da exceção no manipulador de exceções é um supertipo da exceção que foi lançada. Uma vez capturada a exceção, o manipulador que a capturou pode (i) relançar a exceção, (ii) encapsular a exceção ou os dados originados dela em outra exceção, (iii) substituir a exceção capturada por outra, ou (iv) não realizar nenhum processamento com a mesma finalizando, assim, o caminho da exceção. Porém, frequentemente, a exceção que foi levantada não é tratada pela aplicação de forma a permanecer não capturada e isso acontece quando não há nenhum manipulador definido para o tipo de exceção no fluxo.

PUC-Rio - Certificação Digital Nº 0812598/CA

Tabela 16 – Classificação dos fluxos de exceção por aplicação.

	CMS	WordNet	COS-HMS
Sinalizador: Classe			
Não Capturadas	0	2,56097E+14	2,22995E+13
Capturadas			
Específico	126	224	16
Subsunção	7,84544E+13	8,54479E+14	6,70741E+13
Substituídas			
Específico	0	0	0
Subsunção	0	25786	0
Relançamentos			
Específico	0	4541	0
Subsunção	0	44743	0
Encapsuladas			
Específico	0	5861	14422
Subsunção	0	9,5061E+12	669037
Sinalizador: Aspectos			
Não Capturadas	0	3,31676E+13	5,65078E+12
Capturadas			
Específico	31	30	9
Subsunção	6,5263E+13	6,02489E+14	5,41782E+13
Substituídas			
Específico	0	0	0
Subsunção	0	25108	0
Relançamentos			
Específico	0	24478	0
Subsunção	0	70226	
Encapsuladas			0
Específico	0	4550	2112
Subsunção	0	6,70321E+12	563636
Total fluxos	1,44E+14	1,76E+15	1,49E+14

As subseções seguintes visam analisar, em detalhes, os caminhos da exceção apresentados na Tabela 16. Na Seção 4.3.1.1, comparamos a profundidade das ocorrências de exceções lançadas provendo dados sobre em que ponto da aplicação as exceções são lançadas. já na Seção 4.3.1.2, determinamos como as exceções ocorridas em aspectos são tratadas pela aplicação.

PUC-Rio - Certificação Digital Nº 0812598/CA

4.3.1.1.Nível de Profundidade das Exceções Lançadas

Uma dúvida frequente entre os desenvolvedores é a de quem é o responsável pelo lançamento das exceções numa aplicação. Na Figura 45, é apresentado o logaritmo do número de fluxos de exceção por profundidade sinalizando que o maior número de exceções ocorre ao final do fluxo de execução. Para tanto, foi utilizada a escala logarítmica sobre o número de caminhos excepcionais devido à característica do número de fluxos numa aplicação crescer exponencialmente com a profundidade do fluxo. Isso fez-se necessário, pois, caso o gráfico fosse apresentado de outra forma, as informações contidas nos valores iniciais se perderiam.

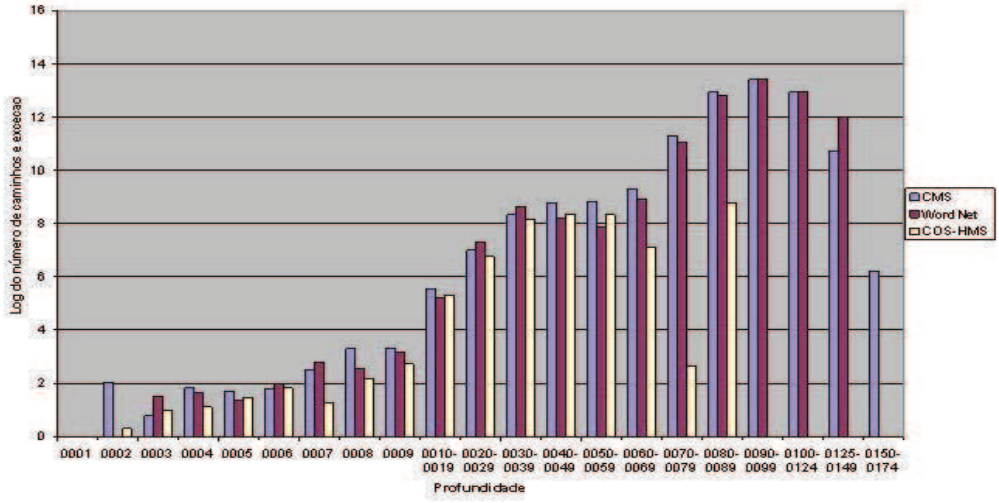

Figura 45 – Número de fluxos de exceção por sistema.

PUC-Rio - Certificação Digital Nº 0812598/CA

A informação de que as exceções são lançadas por método de maior profundidade no fluxo é confirmada pela Figura 46 em que são apresentados os números de instruções que levantam exceções agrupadas pela profundidade do fluxo. Percebe-se, então, que a profundidade em que são encontradas as instruções que sinalizam as exceções, está na metade do fluxo em diante indicando, assim, que os principais pontos onde ocorrem as exceções não fazem parte do código da aplicação e sim das bibliotecas e aspectos participantes da aplicação.

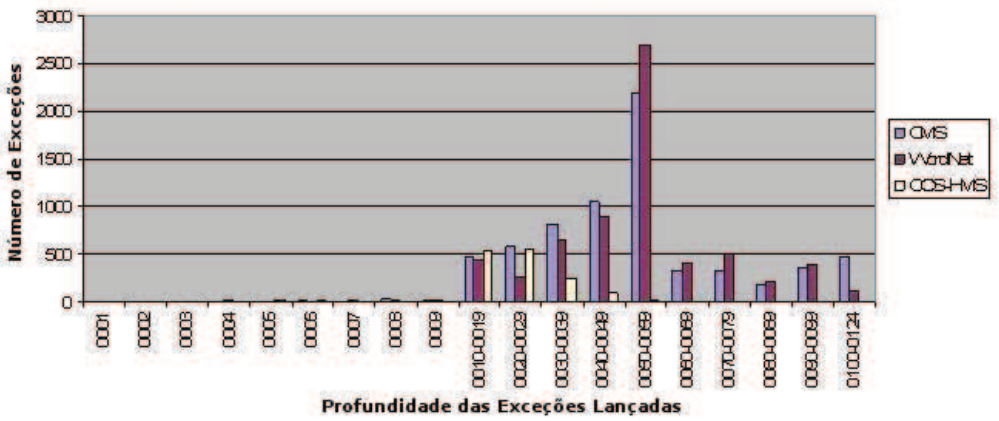

Figura 46 – Número de exceções lançadas por profundidade.

No Anexo 8.1, é apresentado um exemplo de saída da ferramenta com relação aos caminhos de exceção encontrados como o caminho da exceção `java.lang.IndexOutOfBoundsException` na aplicação CMS. Inicialmente, nas linhas 1 e 2 da listagem do Anexo 8.1, vemos que o caminho é iniciado na aplicação CMS e, em seguida, nas linhas 3 a 12, o fluxo ocorre dentro de bibliotecas do servidor de aplicação. Contudo, a chamada da linha 12 para a linha 13 ocorre através do algoritmo CHA (ver Seção 3.2.2.1), uma vez que o algoritmo de point-to não conseguiu determinar a instância do objeto chamado. Assim `org.jboss.aop.joinpoint.ConstructorInvocation` é a classe encontrada que implementa a interface `org.jboss.aop.advice.Interceptor` e, novamente através do algoritmo CHA, ocorre a chamada da linha 13 para a linha 14. Já da linha 14 até a linha 24, ocorrem chamadas relativas ao gerenciamento de transação da aplicação e, a partir da linha 25, ocorrem chamadas relativas ao banco de dados HSQLDB (HSQLDB, 2000), em que ocorre a maior parte do fluxo de exceção.

PUC-Rio - Certificação Digital Nº 0812598/CA

4.3.1.2.Impacto dos Aspectos no Tratamento de Exceção

Determinados comportamentos no tratamento de exceção são prejudiciais à robustez da aplicação, tal como a captura da exceção por subsunção (COELHO, 2008). Com o objetivo de detectar possíveis problemas devido à adição de aspectos nas aplicações selecionadas, analisamos como as exceções são tratadas. A Figura 47 ilustra o número total de caminhos de exceção originados nos aspectos que interceptam a aplicação em que as exceções (i) escaparam, (ii) foram capturadas por subsunção, (iii) foram capturadas por tratadores especializados, (iv) foram capturadas por subsunção e substituídas por outra exceção, (v) foram capturadas por tratadores especializados e substituídas por outra exceção, (vi) foram capturadas por subsunção e em seguida relançadas, (vii) foram capturadas por tratadores especializados e em seguida relançadas, (viii) foram capturadas por subsunção e encapsuladas em outra exceção, (ix) foram capturadas por tratadores especializados e encapsuladas em outra exceção.

Apesar das aplicações analisadas, os métodos que iniciam fluxos têm por padrão capturar todas as exceções que herdam da classe `java.lang.Exception`, mas existem algumas exceções ao padrão e é a partir delas que algumas exceções escapam – excetuando a aplicação CMS, onde rigorosamente todos os métodos inicializadores de fluxo capturam a `java.lang.Exception`. Outro problema encontrado nas aplicações é que a maioria das exceções lançadas por aspectos é capturada por subsunção, levando a aplicação a tratar de forma inadequada os fluxos excepcionais. Uma vez que os problemas acima foram identificados nas aplicações analisadas confirma-se a hipótese nula (H0) do nosso estudo.

Nas aplicações CMS e COS-HMS não há relançamento e nem substituição de exceções, porém, na aplicação WordNet, tanto ocorre o relançamento quanto a substituição de exceções. Assim, em relação ao encapsulamento de exceções, somente as aplicações COS-HMS e WordNet utilizam o recurso no seu tratamento de exceção. Portanto, observa-se que a captura por subsunção é fortemente usada para relançamento, substituição e encapsulamento de exceções.

PUC-Rio - Certificação Digital Nº 0812598/CA

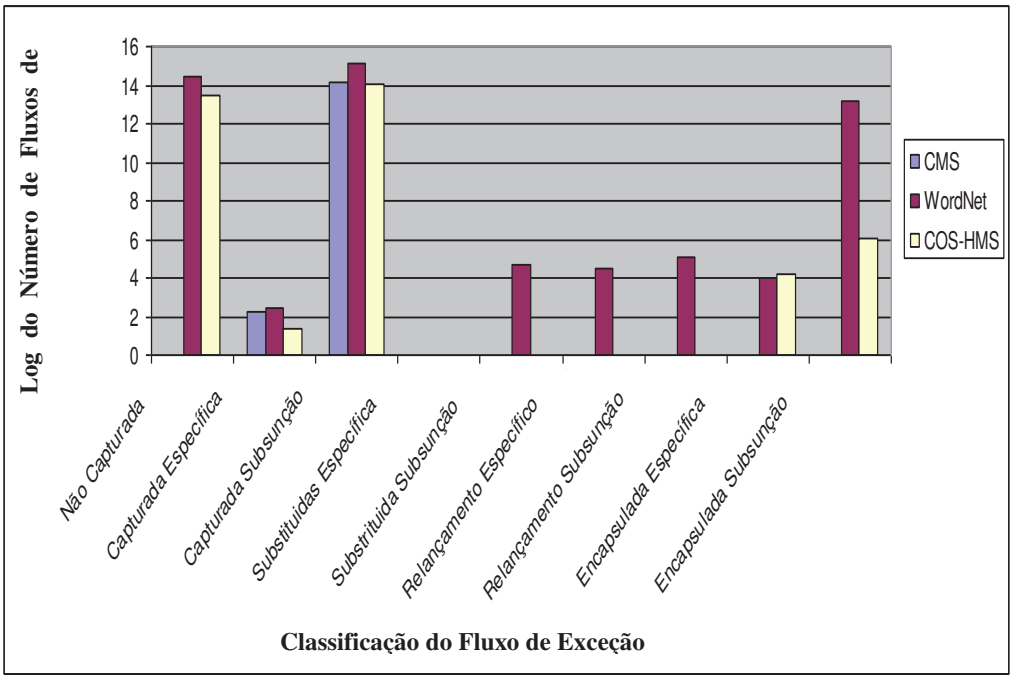

Figura 47 – Tratamento de exceção de exceções originadas em aspectos.

4.3.1.3.A Responsabilidade de Cada Aspecto no Lançamento de Exceções

O framework EJB possui alguns interesses transversais que vão desde a segurança até o gerenciamento de transações. Contudo, alguns destes interesses são compulsoriamente adicionados à aplicação e causam impacto no tratamento de exceção da aplicação.

Na Tabela 17, são encontrados os números de cláusulas `throw` no fluxo iniciado por cada aspecto que intercepta as aplicações, sendo que todas estas são interceptadas pelos mesmos aspectos, já que utilizam o framework EJB da mesma forma. Cada aspecto na listagem abaixo possui responsabilidades, sendo elas: (i) `org.jboss.aspects.tx.TxPropagationInterceptor`, tem por responsabilidade propagar uma exceção de um contexto de execução para outro; (ii) `org.jboss.ejb3.asynchronous.AsynchronousInterceptor`, tem por responsabilidade tornar transparente a chamada a métodos assíncronos; (iii) `org.jboss.ejb3.ENCPropagationInterceptor`, propaga o contexto de execução; (iv) `org.jboss.ejb3.stateless.StatelessInstanceInterceptor`, gerencia o pool de instâncias de Stateless Session Beans.

PUC-Rio - Certificação Digital Nº 0812598/CA

Assim, analisando as aplicações, percebe-se que nenhuma chamada acessa métodos de forma assíncrona, mas a execução do aspecto `AsynchronousInterceptor` ocorre nas aplicações analisadas. Todavia, a execução dos métodos que levantam exceções depende de informações contidas no contexto de exceção do aspecto (ver Seção 2.1.1). Posto isto, a hipótese (H1) é refutada, ou seja, os aspectos analisados são necessários para a execução das aplicações citadas e, por outro lado, esta análise dos aspectos aponta a falta de precisão, no que diz respeito aos fluxos de exceção, que a ferramenta possui.

Tabela 17 – Número de fluxos de exceção por aspecto.

Classe do Aspecto	Número de fluxos de Exceção
`org.jboss.aspects.tx.TxPropagationInterceptor`	34
`org.jboss.ejb3.asynchronous.AsynchronousInterceptor`	4,87 E+12
`org.jboss.ejb3.ENCPropagationInterceptor`	0
`org.jboss.ejb3.stateless.StatelessInstanceInterceptor`	5,35 E+10

PUC-Rio - Certificação Digital Nº 0812598/CA

4.3.1.4.Tipo das Exceções

O tipo de uma exceção esta diretamente associada ao tipo da falha, por exmeplo: o lançamento de `java.lang.RuntimeException` indica falhas de programação, enquanto `java.lang.Error` indica problemas na máquina virtual (ver Seção 2.2). Desta forma, o tipo das exceções e o número de fluxos que esta exceção pode ser levantada, determinam o grau de importância que o desenvolvedor deve dar para cada exceção. As listagens das exceções de cada aplicação podem ser encontradas nos anexos.

Nas aplicações analisadas, percebe-se um número elevado de tipos exceção que são levantadas: 84 para Health Care (ver Anexo 8.4), 150 para WordNet (ver Anexo 8.3) e 142 para CMS (ver Anexo 8.2). Destas, as exceções de Ruintime possuem o maior número de fluxos de exceção, e assim, o número de exceções checadas é da ordem 100 vezes menor e as exceções do tipo `java.lang.Error` são inxistentes ou insignificantes. Dentre as exceções lançadas, temos como destaque a exceção `java.lang.RuntimeException`, a qual possui o maior

número de fluxos de exceção, onde a causa principal deste número excessivo de fluxos está no JBoss AS que encapsula várias de suas exceções numa `RuntimeException`, isto será discutido mais adiante na Seção 4.4.1.

4.4.Padrões de Tratamento de Exceção Encontrados

Inspecionando as aplicações estudadas, não foi encontrado nenhum dos padrões de erros identificados por Coelho (2008), acreditamos que este fato ocorreu devido aos aspectos das aplicações que estudamos serem providos pelo framework EJB, cujos interesses estão relacionados ao gerenciamento de transações, persistência, distribuição, dentre outros (ver Seção 2.3.1). Cada um dos aspectos analisados (ver **Erro! Fonte de referência não encontrada.**) foram inspecionandos manualmente e nenhum deles captura exceções das classes que forem interceptadas. As seções seguintes apresentam os padrões encontrados.

4.4.1.Encapsulamento de Exceções em RuntimeException

Encapsular exceções em outra exceção é uma prática comum no desenvolvimento de aplicações. Contudo, o objetivo do encapsulamento é adicionar semântica a exceção ou forçar que a aplicação obedeça a uma interface de exceções (ver Seção 3.2.4.1).

O encapsulamento de exceções em RuntimeExceptions foi identificado a partir de um número elevado de encapsulamentos numa determinada profundidade de código, e em seguida, foi realizada a analise dos fluxos de exceções originados na profundidade encontrada. A partir desta análise, foi identificado que o encapsulamento tanto é feito para exceções checadas quanto para a exceção `java.lang.Throwable`. As consequências de cada encapsulamento serão discutidas a seguir.

PUC-Rio - Certificação Digital Nº 0812598/CA

```
public InitialContext getInitialContext(){
 try{
 return
InitialContextFactory.getInitialContext(initialContextProperties);
 } catch (NamingException e){
     throw new RuntimeException(e);
 }
}
```

Figura 48 – Exemplo de encapsulamento de exceções checadas em exceção não-checada na classe org.jboss.ejb3.EJBContainer.

Exceções não-checadas indicam falhas de programação (GOODENOUGH, 1975a), então encapsular uma exceção checada numa exceção não-checada entende-se que a exceção que foi encapsulada representa uma falha no desenvolvimento para a classe ou biblioteca que a capturou. Na Figura 48 é apresentado um exemplo de tratamento de exceção cujo objetivo é evitar que a interface de exceções do método `getInitialContext` seja modificada e desta forma, ocultando do desenvolvedor a necessidade de tratar a exceção. Esta forma de tratamento de exceção é semelhante à técnica utilizada para lançamento da exceção `SoftException`, a qual é utilizada no AspectJ para que exceções checadas possam ser levantadas mesmo que os métodos interceptados não as declare na clausula `throws` (COELHO, 2008).

PUC-Rio - Certificação Digital Nº 0812598/CA

```
public Object construct(){
Interceptor[] cInterceptors;
cInterceptors                                                    =
constructorInterceptors[defaultConstructorIndex];
     …
ConstructorInvocation invocation;
invocation = new ConstructorInvocation(cInterceptors);
     …
try{
   return invocation.invokeNext();
}catch (Throwable throwable) {
    throw new RuntimeException(throwable);
}
}
```

Figura 49 – Exemplo de encapsulamento da exceção Throwable em exceção não-checada na classe org.jboss.ejb3.EJBContainer.

A classe `java.lang.Throwable` é o supertipo de todas as exceções, incluindo as classes `java.lang.Error`, e estas indicam falhas graves, e normalmente sinalizam para a máquina virtual que deve ser executada alguma ação emergencial, como por exemplo: executar a rotina de garbage collector para liberar mais memória para a aplicação (ver Seção 2.2). Então, uma vez que exceções do tipo `java.lang.Throwable` e suas subclasses são capturadas, a máquina virtual não receberá sinais que indicam um problema, o qual ela deveria tomar ações para reestabelecer seu funcionamento normal. Pode ser visto na Figura 49, um exemplo de como o servidor de aplicação JBoss AS encapsula `java.lang.Throwable` em `java.lang.RuntimeException`.

4.4.2.Captura de java.lang.Exception na Camada de Apresentação

No processo de análise do tratamento de exceção das aplicações foi identificado que na camada de apresentação as exceções são capturadas por `java.lang.Exception`. Apesar de capturar `java.lang.Exception` não ser considerada uma boa prática para o tratamento de exceções, este padrão tem-se difundido em aplicações Web.

PUC-Rio - Certificação Digital Nº 0812598/CA

Como foi visto na Seção 0, existem muitos tipos de exceções que são lançados numa aplicação e maioria delas é subtipo de `RuntimeException`. Desta forma, não é razoável para o desenvolvedor gerar códigos que capture individualmente cada exceção, e assim a maneira mais objetiva é capturar `java.lang.Exception`. Acreditamos que esta é a explicação para o número elevado de exceções capturadas por subsunção nas aplicações.

4.5.Restrições do Estudo

O estudo foi realizado em aplicações corporativas que utilizam o framework EJB com o objetivo de realizar análises em aplicações realistas, mas não foi encontrada nenhuma aplicação relevante que utilize o JBoss AOP de forma isolada. Desta forma, apesar dos domínios e arquitetura diferentes, os resultados das análises são semelhantes para as aplicações analisadas.

Outra restrição do estudo foi a remoção de classes de bibliotecas que geraram caminhos impossíveis de ocorrer, como, por exemplo a biblioteca Struts2, que possui a classe `BeanAdapter` que implementa a interface `Element`. O problema ocorre quando a ferramenta analisa bibliotecas responsáveis pelo carregamento dos arquivos de configuração do JBoss AOP, e esta biblioteca processa o xml utilizando a interface `Element`. Assim, quando a ferramenta analisa um método da interface `Element`, a mesma irá buscar todas as classes que implementam esta interface e, a partir de então, a classe `BeanAdapter` passa a fazer parte do fluxo excepcional.

4.6.Resumo

Este estudo foi realizado para avaliar o impacto que os aspectos do framework EJB implementados com o JBoss AOP tem sobre o tratamento de exceções das aplicações CMS, Wordnet e COS-HMS.

Estes sistemas representam domínios de aplicação diferentes além de possuírem arquiteturas distintas das utilizadas pelo mercado, de tal forma que representam cenários realistas para a validação da ferramenta. As três aplicações apresentam comportamento semelhantes em relação ao tratamento de exceção, pois as exceções são capturadas nas classes que iniciam o fluxo da requisição do

PUC-Rio - Certificação Digital Nº 0812598/CA

usuário. Acreditamos, portanto, que esta estrutura de tratamento de exceção adotada pelas aplicações seja um padrão utilizado pelo mercado a fim de evitar mensagens indesejadas para o usuário – sendo estas entendidas como mensagens não formatadas pela aplicação, mesmo que estejam incorretas, o que frequentemente ocorre nas aplicações analisadas devido à maneira que as exceções são tratadas nas aplicações. Assim, a análise das aplicações indicou que alguns fluxos de exceção ocorrem devido a aspectos adicionados pelo framework EJB, mas alguns destes aspectos não são utilizados pelas aplicações analisadas, de forma que estes não utilizados adicionam complexidade ao tratamento de exceção e não trazem nenhum benefício para ela.

PUC-Rio - Certificação Digital Nº 0812598/CA