Programação Orientada a Aspectos
Introdução a conceitos teóricos e práticos

Adriano G. do Prado[1], José E. M. Lemos[1], José F. da S. Cruz[1], Thales E. Nazatto[1]

[1]Departamento de Estatística, Matemática Aplicada e Computação (DEMAC)
Universidade Julio de Mesquita Filho (Unesp – Rio Claro)

{adriano.prado, jose.edu.lemos, jfernandocruz.lp, tenazatto}@gmail.com

Abstract. *The aspect-oriented programming appears on the scene as a computational alternative paradigm for solving classic problems of abstraction and modularization of systems addressed by structural programming and also the object-oriented. This short article attempts to elucidate its main theoretical concepts and show features of the implementation using Aspect J.*

Resumo. *A programação orientada a aspectos aparece no cenário computacional como um paradigma alternativo para o saneamento de problemas clássicos de abstração e modularização de sistemas, abordados pela programação estrutural e também a orientada a objetos. Este pequeno artigo tenta elucidar seus principais conceitos teóricos e apresentar características de sua implementação utilizando Aspect J.*

1. Introdução

Durante os ultimos anos temos acompanhado a crescente evolução e adoção da programação orientada a objetos (POO). Esta veio com o objetivo de responder às questões essenciais e inerentes ao desenvolvimento de software no que diz respeito especialmente às facilidades e manutenção do código.

A orientação a objetos veio como forma de sanar uma das deficiências do desenvolvimento estruturado. Apesar de interesses relativos às funcionalidades ficarem separados, interesses relativos a dados ficavam distribuídos em diversos módulos, o paradigma orientado a objetos definiu que a separação deveria acontecer em duas dimensões, primeira dividida em termos de dados e depois em termos das funções que utilizam cada tipo de dados.

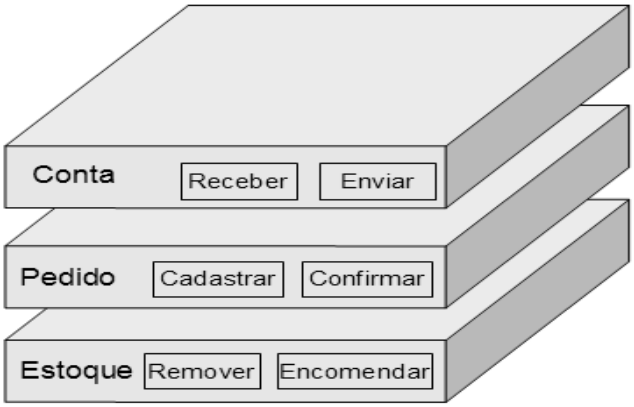

Figura 1. Exemplo de divisão dos interesses em Programação Orientada a Objetos.

As linguagens atuais de programação como Java, por exemplo, fornecem condições para representação abstrata de funções, métodos ou classes etc. Mas como tratar aspectos que não são inerentes ao negócio no sistema? Como tratamento de exceções, log de sistemas e concorrência? Nesse contexto a Programação Orientada a Aspectos é apresentada a seguir.

2. Programação Orientada a Aspectos

A programação orientada a aspectos (AOP – Aspect Oriented Programming), foi criada em Palo Alto nos laboratórios da Xerox (Kiczales , 1997). Segundo [Lopes 1997], “as aplicações estão ampliando os limites das técnicas de programação atuais, de modo que certas características de um sistema afetam seu comportamento de tal forma que as técnicas atuais não conseguem capturar essas propriedades de forma satisfatória.”.

Na concepção de Soares e Borba (2002), “a programação orientada a aspectos propõe não apenas uma decomposição funcional, mas também ortogonal do problema". POA “permite que a implementação de um sistema seja separada em requisitos funcionais e não funcionais.”.

Segundo [Piveta 2001], “Quando duas propriedades sendo programadas devem ser compostas de maneira diferente e ainda se coordenarem é dito que elas são ortogonais entre si”, conhecidos como “*interesses ortogonais” ou “interesse entrecortante” (crosscuting concerns).*

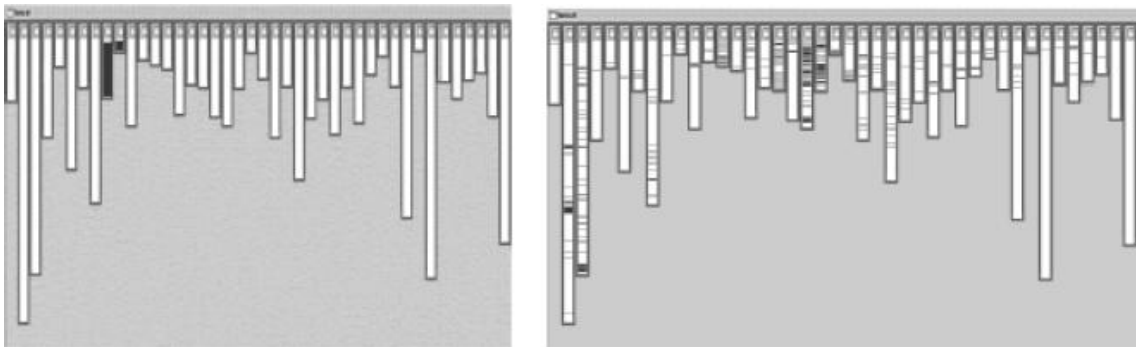

Figura 2. Exemplo de interesses ortogonais.

Neste sentido podemos afirmar que o objetivo principal da orientação a aspectos é encapsular interesses entrecortantes em módulos fisicamente separados do restante do código. Esses módulos são conhecidos como *aspectos*. Pensando em termos abstratos, a orientação a aspectos introduz uma terceira dimensão de decomposição. Além de decompor o sistema em objetos (dados) e métodos (funções), decompomos cada objeto e função de acordo com o interesse sendo servido e agrupamos cada interesse em um módulo distinto, ou aspecto.

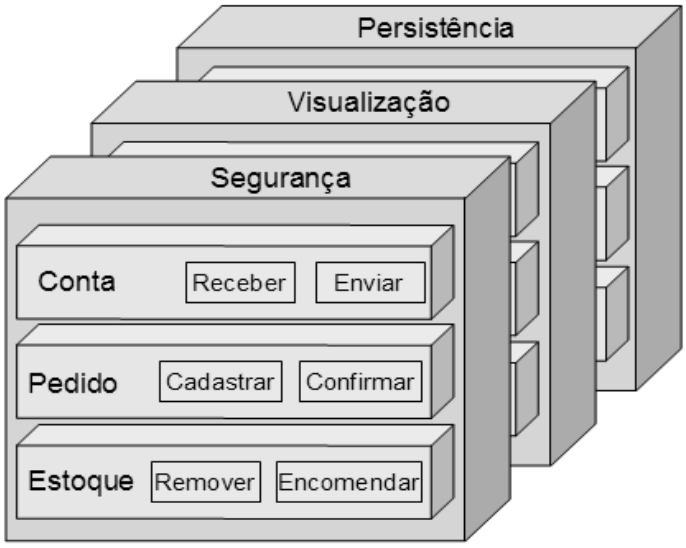

Figura 3. Decomposição da POA.

A composição da POA consiste nos seguintes componentes: linguagem de componentes, linguagem de aspectos e combinador de aspectos; detalhados a seguir:

- *Linguagem de componentes*: são programados os códigos bases, segundo [Irwin 1997] “a linguagem de componentes deve permitir ao programador escrever programas que implementem as funcionalidades básicas do sistema, ao mesmo tempo em que não prevêem nada a respeito do que deve ser implementado na linguagem de aspectos”.

- *Linguagem de aspectos*: na qual se programam os aspectos . A linguagem de aspectos deve suportar a implementação das propriedades desejadas de forma clara e concisa, fornecendo construções necessárias para que o programador crie estruturas que descrevam o comportamento dos aspectos e definam em que situações eles ocorrem.

- *Combina dor de aspectos*: combinar os programas escritos em linguagem de componentes e os escritos em linguagem de aspectos, segundo [Neves 2009] "O combinador de aspectos processa os componentes e a linguagem de aspectos, compondo-os para produzir a funcionalidade desejada do sistema. (...) o processo que une os componentes e os aspectos em um programa executável é chamado de combinação (*weaving*) . A combinação dos aspectos pode ser realizada de forma estática ou dinâmica".

3. AspectJ

Atualmente existem diversas opções para utilização da programação orientada a aspectos, este artigo abordará a utilização do *AspectJ* como linguagem de aspectos, a linguagem Java como linguagem de componentes e a plataforma Eclipse como ambiente de desenvolvimento. Ambos são distribuídos gratuitamente e estão disponíveis em diversas plataformas. Segundo *Kiczales* (2001), "O *AspectJ* é uma extensão simples e prática da AOP para Java (*Kiczales,* 2001)".

No paradigma de programação orientada a aspectos há quatro conceitos elementares: Aspecto (*Aspect*), Ponto de Junção (*Join Point*), Ponto de Atuação (*Pointcut*) e Adendo (*Advice*).

Aspectos: São unidades de código que encapsulam diversos pontos de atuação e adendos de um programa. Sua utilidade e funcionamento se assemelham a de uma classe, porém sua disposição no código diferencia-se pela maior modularização.

Pontos de Junção: são pontos bem definidos da execução de um programa, por exemplo, uma chamada a um método ou a ocorrência de uma exceção, entre muitos outros.

Pontos de Atuação: são elementos do programa usados para definir um *join point*, como uma espécie de regra criada pelo programador para especificar eventos que serão atribuídos a *join points*. Os *pointcuts* existem para que possam ser criadas regras genéricas para definir os eventos que serão considerados "*join points*", para que não seja necessário definir cada *join point* individualmente (o que tornaria a POA quase sem sentido). Outra função dos *pointcuts*, além de definir os eventos considerados como *join points*, é apresentar dados do contexto de execução de cada *join point*, que serão utilizados pela rotina disparada pela ocorrência do *join point* em questão.

Adendos são pedaços da implementação de um aspecto que são executados em pontos bem definidos da execução do programa principal (*join points*). Eles são compostos de duas partes : a primeira delas é o *pointcut*, que define as regras de capturados *join points*, e a segunda delas são o código que será executado.

A execução desses conceitos é exemplificada nas figuras a seguir.

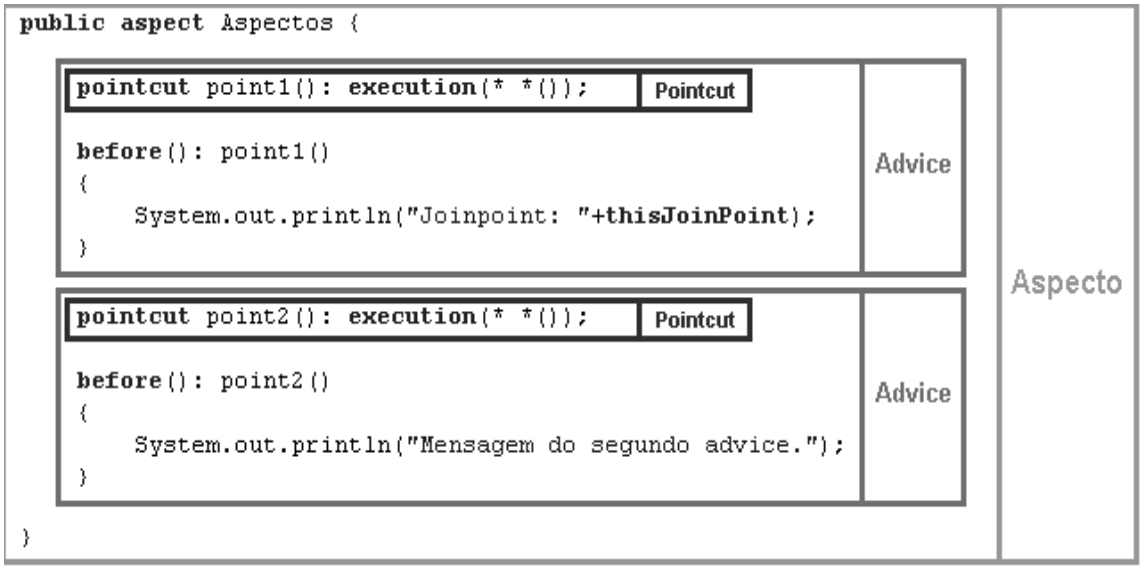

Figura 4. Exemplo dos conceitos de Aspecto.

4. Conclusão

A programação orientada a aspectos possibilita que o programador separe o código referente ao negócio dos interesses ortogonais, centralizando estes interesses em aspectos bem definidos no sistema.

Esta separação propiciada pela POA (programação orientada a aspectos) pode trazer diversos benefícios para uma equipe de desenvolvimento. Para todos os sistemas, poderia ser utilizado um mesmo programador especialista no tratamento de exceções, sistema de log, conexão com banco de dados, etc, já que não há necessidade que este programador conheça o código referente ao negócio do sistema.

Referências

Neves, Vânia de Oliveira , O. (2009) “Teste de integração contextual de programas orientados a objetos e aspectos” ;

Andrade, C., Goetten, V. e Winck, Diogo. (1995) “Programação orientada a Aspectos abordando Java e AspectJ”, http://inf.unisul.br/~ines/workcomp/cd/pdfs/2337.pdf

Andrade, C., Goetten, V. e Winck, Diogo. (2006) “AspectJ - Programação Orientada a Aspectos com Java”, Editora Novatec.

Kixzales, G. , Lamping, J., Mendhekar, A., Maeda , Crhis.,Videira, C., Irwin,J. (1997), ”Aspect - Oriented Programming” http://cseweb.ucsd.edu/users/wgg/CSE218/aop-ecoop97.pdf