UTILIZAÇÃO DO PARADIGMA DE PROGRAMAÇÃO ORIENTADO A ASPECTO NA OTIMIZAÇÃO DO DESENVOLVIMENTO DE SOFWARES

Rafael Alessandro CASACHI[1], **Almir Rogério CAMOLESI**[2]

[1] Departamento de Informática - Instituto Municipal de Ensino Superior de Assis – Fundação Educacional do Município de Assis (FEMA) - Assis - SP – Brasil

[2] Departamento de Informática - Instituto Municipal de Ensino Superior de Assis – Fundação Educacional do Município de Assis (FEMA) - Assis - SP – Brasil

rcasachi@gmail.com, camolesi@fema.edu.br

A necessidade de uma tecnologia que pudesse otimizar a produção de software estimulou a equipe de cientistas do PARC (*Palo Alto Research Center*) da Xerox liderados por Gregor Kiczales a desenvolver um paradigma que pudesse separar os códigos de acordo com o seu comportamento.

Este trabalho visa mostrar como a separação destes comportamentos e a utilização do paradigma de programação orientada a aspecto (POA) pode simplificar a produção de códigos, tornando-os mais organizados e eficientes.

Os códigos principais do sistema, chamados funcionais são separados dos códigos sistêmicos, que são aqueles códigos que oferecem suporte ao código funcional. Com está separação, pode-se implementar as aplicações separadamente unindo as funcionalidades através do combinador.

O combinador é o compilador da POA, com ele pode-se unir um código sistêmico ao código funcional, ou seja, pode-se unir um aspecto ao ponto de junção no código principal. Para que esta união seja bem-sucedida a definição de dois elementos no aspecto é essencial: o ponto de atuação e o adendo.

Um ponto de atuação contém a assinatura do ponto de junção que o aspecto irá ser combinado. O adendo é o bloco de código que será executado no ponto de junção. O momento de execução de um aspecto é configurado dentro do adendo, através das palavras-chaves: *before*, *after* e *around*.

A implantação da POA evita que os programas sejam implementados contendo dois problemas comuns no desenvolvimento de softwares: o código espalhado e o código emaranhado. O código espalhado ocorre quando um interesse é espalhado em

vários módulos, métodos ou classes e o código emaranhado é quando existe uma classe, módulo ou método programado com vários interesses. Este tipo de problema prejudica a visibilidade do código, dificultando uma futura manutenção ou implantação.

Referências

[1] BODKIN, Ron; LADDAD, Ramnivas. Zen and the art of Aspect-Oriented Programming. Linux Magazine, April, 2004.

[2] GOETTEN, Vicente J.; WINCK, Diogo V. AspectJ – Programação Orientada a Aspectos com Java. São Paulo: Novatec Editora, 2006.

[3] KICZALES, Gregor; LAMPING, John; MENDHEKAR, Anurag; MAEDA, Chris; LOPES, Cristina Videira; LOINGTIER, Jean-Marc. Aspect-Oriented Programming. In: European Conference on Object-Oriented Programming (ECOOP), 06,1997. Finlândia. Anais Springer-Verlag LNCS 1241, 06, 1997

[4] SAFONOV, Vladimir O. Using Aspect-Oriented Programming for Trustwothy Software Development. New Jersey: Wiley-Interscience, 2008.

[5] THE ASPECT TEAM. The AspectJ Programming Guide. Xerox Corporation, Palo Alto Research Center (PARC), Palo Alto, CA, Estados Unidos. Disponível em: <”www.eclipse.org/aspectJ”>. Acessado em: 06 abr. 2011.