Desenvolvimento de Software Orientados a Aspectos Utilizando RUP e AspectJ

Sérgio Soares[1]*, Paulo Borba[1]

[1]Centro de Informática, Universidade Federal de Pernambuco
Av. Prof. Luiz Freire S/N, Caixa Postal 7851,
CEP 50740-540 Recife, PE
e-mail: [scbs,phmb]@cin.ufpe.br

Resumo

Atualmente muito se fala em Programação Orientada a Aspectos (AOP — Aspect-Oriented Programming). AOP procura solucionar algumas ineficiências da orientação a objetos, como o entrelaçamento e espalhamento de código com diferentes propósitos. Este entrelaçamento e espalhamento tornam o desenvolvimento e a manutenção destes sistemas extremamente difícil. AOP aumenta a modularidade separando código que implementa funções específicas, afetando diferentes partes do sistema, chamadas preocupações ortogonais (*crosscutting concern*). Na direção do desenvolvimento de software orientado a aspectos (AOSD — Aspect-Oriented Software Development), demonstramos neste tutorial como integrar o RUP, um processo de desenvolvimento de software muito conhecido e utilizado nas empresas de desenvolvimento, com um método de implementação que dá suporte a modularização de requisitos presentes na maioria dos softwares atuais. Tais requisitos são gerenciamento de dados, distribuição e controle de concorrência. O método fornece um framework de aspectos para a implementação de tais requisitos além de contar com um suporte automatizado para a geração de aspectos específicos da aplicação com AspectJ, uma extensão orientada a aspectos da linguagem Java.

Abstract

Aspect-Oriented Programming (AOP) is being quite commented nowadays. AOP tries to solve some inefficiencies of the Object-Oriented Programming, such as spread and tangled code with different purposes. This spread and tangled code makes development and maintenance hard. AOP increases software modularity by separating code that implements specific functions affecting several parts of the software, called crosscutting concern. Toward Aspect-Oriented Software Development (AOSD), this tutorial addresses how to integrate RUP, a well-known and used software development process, with an implementation method that supports modularization of some requirements present in most software. These requirements are data management, distribution, and concurrency control. The implementation method has an aspect framework, to implement these crosscutting concerns, and tool support to generate application specific aspects with AspectJ, an aspect-oriented extension of Java.

1. Desenvolvimento Orientado a Aspectos

A necessidade de software de qualidade motivou o uso da programação orientada a objetos [1] em busca de maiores níveis de reuso e manutenibilidade, aumentando a produtividade e o suporte à mudanças de requisito. Porém, o paradigma orientado a objetos apresenta

*Apoiado pela CAPES

limitações [10, 11], tais como o entrelaçamento e o espalhamento de código através de vários requisitos. Por exemplo, código de distribuiçào entrelaçado com código de negócio e de interface com o usuário e código de controle de concorrência espalhado por diversos módulos do software.

Desta forma, é cada vez mais notada a vantagem e, portanto, a necessidade do uso da programação orientada a aspectos (AOP) [3, 6], que supera estas deficiências da orientação a objetos. Vários pesquisadores, inclusive nós, acreditam que o Desenvolvimento Orientado a Aspectos é bastante promissor [9, 3, 7].

Este tutorial apresenta exemplos concretos de problemas da orientação a objetos em aplicações práticas e reais, além dos passos que devem ser seguidos em um desenvolvimento orientado a aspectos. Também daremos exemplos de vários *crosscutting concerns*, em especial dos identificados no sistema utilizado como exemplo. O tutorial também faz uma breve introdução à linguagem de implementação AspectJ [5, 13], uma extensão orientada a aspectos da linguagem Java, apresentando algumas construções, bem como exemplos do seu uso demonstrando a expressividade da mesma.

Um ponto importante é esclarecer que AOP não dispensa análise e projeto de software, como alguns podem imaginar a primeira vista, através do relato da nossa experiência ao implementar um sistema com AOP.

2. Método de Implementação Orientado a Aspectos

Como a maioria dos paradigmas, a Orientação a Aspectos surgiu com uma linguagem de programação e com atividades de implementação. Neste tutorial apresentamos um método de implementação orientado a aspectos [12, 16] e uma arquitetura [8] que pode ser utilizada para implementar vários tipos de sistemas. O método possui um framework de aspectos [15] que fornece implementação de requisitos de gerenciamento de dados, distribuição e controle de concorrência, que além de permitir seu reuso, fornecem um guia para o programador e permitem a criação de ferramentas de geração de código que dão suporte ao método. O framework está implementado em AspectJ.

Além disso, o método de implementação permite a adoção de uma abordagem de implementação progressiva [14, 2] que permite uma maior produtividade de implementação. Isto acontece devido a uma antecipação de eventuais mudanças de requisitos, sem que tenha sido empregado esforço para implementar requisitos de persistência, distribuição e de controle de concorrência, evitando, portanto, que em decorrência de mudanças de requisitos funcionais estes também tenham de ser modificados.

3. Integração com o RUP

Por fim, demonstramos como um conhecido e bastante utilizado processo para desenvolvimento de software, RUP [4], pode ser adaptado para utilizar o método de implementação em questão. Apresentamos que atividades de gerenciamento, requisitos, análise, projeto e implementação são afetadas pelo método de implementação, bem como novas atividades que devem ser adicionadas ao processo para a correta execução do método, indo na direção do Desenvolvimento de software Orientado a Aspectos.

Além disso, discutimos mudanças na estrutura dinâmica do RUP devido à adoção do método de implementação e da abordagem progressiva. Vale salientar que a adoção da abordagem progressiva não é obrigatória, nos levando a discutir alternativas no uso da mesma.

Referências

[1] Grady Booch. *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings, second edition, 1994.

[2] Paulo Borba, Saulo Araújo, Hednilson Bezerra, Marconi Lima, and Sérgio Soares. Progressive Implementation of Distributed Java Applications. In *Engineering Distributed Objects Workshop, ACM International Conference on Software Engineering*, pages 40–47, Los Angeles, EUA, 17th–18th May 1999.

[3] Tzilla Elrad, Robert E. Filman, and Atef Bader. Aspect–Oriented Programming. *Communications of the ACM*, 44(10):29–32, October 2001.

[4] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

[5] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G. Griswold. Getting Started with AspectJ. *Communications of the ACM*, 44(10):59–65, October 2001.

[6] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. In *European Conference on Object-Oriented Programming, ECOOP'97*, LNCS 1241, pages 220–242, Finland, June 1997. Springer-Verlag.

[7] Ramnivas Laddad. I want my AOP!, Part 1: Separate software concerns with aspect-oriented programming. *JavaWorld*, January 2002. Avaliable at http://www.javaworld.-com/javaworld/jw-01-2002/jw-0118-aspect.html.

[8] Tiago Massoni, Vander Alves, Sérgio Soares, and Paulo Borba. PDC: Persistent Data Collections pattern. In *First Latin American Conference on Pattern Languages of Programming — SugarLoafPLoP. Published in University of São Paulo Magazine — ICMC, 2002*, pages 311–326, Rio de Janeiro, Brazil, October 2001.

[9] Gail C. Murphy, Robert J. Walker, Elisa L.A. Baniassad, Martin P. Robillard, Albert Lai, and Milk A. Kersten. Does aspect–oriented programming work? *Communications of the ACM*, 44(10):75–77, October 2001.

[10] H. Ossher, M. Kaplan, A. Katz, W. Harrison, and V. Kruskal. Specifying suject–oriented composition. *TAPOS*, 2(3):179–202, 1996. Special Issue on Subjectivity in OO Systems.

[11] Harold Ossher and Peri Tarr. Using subject–oriented programming to overcome common problems in object–oriented software development/evolution. In *International Conference on Software Engineering, ICSE'99*, pages 698–688. ACM, 1999.

[12] Sérgio Soares. *An Aspect-Oriented Implementation Method*. Tese de doutorado, Centro de Informática, Universidade Federal de Pernambuco, Setembro 2004. A ser publicada.

[13] Sérgio Soares and Paulo Borba. AspectJ — Programação Orientada a Aspectos em Java. In *Tutorial no VI Simpósio Brasileiro de Linguagens de Programação*, pages 39–55, Rio de Janeiro, Brasil, 5 a 7 de Junho 2002.

[14] Sérgio Soares and Paulo Borba. PIP: Progressive Implementation Pattern. In *SDPP'02 - 1st Workshop on Software Development Process Patterns, 17th ACM Conference on Object-Oriented programming systems, languages, and applications, OOPSLA'02*, Technical Report TUM-I0213, Munich University of Technology, Munich 12/2003, November 2002.

[15] Sérgio Soares and Paulo Borba. An aspect-oriented implementation method. *Student Research Extravaganza (Poster Session), International Conference on Aspect-Oriented Software Development, AOSD 2004*. http://www.aosd.net/2004/extravaganza.php. Lancaster, UK, March 2004.

[16] Sérgio Soares, Eduardo Laureano, and Paulo Borba. Implementing Distribution and Persistence Aspects with AspectJ. In *Proceedings of the 17th ACM conference on Object-oriented programming, systems, languages, and applications, OOPSLA'02*, pages 174–190. ACM Press, November 2002. ACM SIGPLAN Notices 37(11).