MAC5715 - Tópicos Avançados de POO
Professor: Fabio Kon

The Adaptive Pipeline Aspect Pattern

Raoni Kulesza e Eduardo Oliveira de Souza

23 de outubro de 2006

1. Objetivos

O padrão *Adaptive Pipeline Aspect* tem como objetivo permitir o desenvolvimento de aplicações adaptativas que se baseiam no padrão arquitetural *Pipes and Filters* e concomitantemente separar os interesses de adaptação das funcionalidades básicas da aplicação, provendo modularidade e manutenibilidade.

2. Motivação

A convergência da Internet com outras redes (telefonia móvel e televisão digital), o surgimento da computação ubíqua [1] e a crescente demanda pela computação autônoma[2], têm favorecido a existência de um ambiente constantemente modificado e altamente dinâmico. Tal cenário exige o desenvolvimento de sistemas de software adaptativos, ou seja, que tenham a capacidade de se adaptar dinamicamente em resposta a mudanças no contexto em que estão inseridos, de forma a: (i) atender mudanças de requisitos; (ii) otimizar o uso de recursos; e (iii) garantir uma qualidade de serviço mínima na sua utilização.

O padrão *Pipes and Filters* [3] descreve um arquitetura de software constituída de componentes que realizam algum tratamento de dados (*filters*) e conexões que transmitem dados entre esses componentes (*pipes*), formando uma cadeia de processos de fluxos de dados (*pipeline*). Tal solução é utilizada freqüentemente em sistemas que tratam fluxos de dados seqüencial e incrementalmente, tais como protocolos de comunicação e sistemas multimídia. O emprego do padrão permite reutilização e manutenibilidade, uma vez que etapas de um *pipeline* podem ser facilmente identificadas e separadas em componentes independentes e que podem colaborar entre si. Entretanto, a adição de requisitos de adaptabilidade (por exemplo, reconfiguracão dinâmica no arranjo de componentes) não tem sido bem tratada nessas arquiteturas, ocasionando a adição de problemas de manutenibilidade [4].

3. Problema

Como obter legibilidade e facilidade de manutenção do código em requisitos de adaptabilidade para sistemas baseados em *Pipes and Filters*.

4. Forças

O padrão proposto considera as seguintes forças:

- Deve existir uma separação modular entre os interesses relacionados à adaptação e os

outros interesses da aplicação-base;
- As funcionalidades de adaptação devem ser facilmente acopladas/desacopladas e ativadas/desativadas, de forma a possibilitar uma substituição, atualização ou remoção dessas funcionalidades;
- As funcionalidades de adaptação devem ter entendimento e manutenção fácil;
- O padrão deve ser genérico de modo a ser empregado no desenvolvimento de aplicações para qualquer plataforma de execução ou cenário de utilização (domínio) com *pipelines*.

5. Solução

Utilizar programação orientada a aspectos (POA) para separar os interesses de adaptação das demais funcionalidades da aplicação. Os aspectos devem interagir direta e/ou indiretamente com qualquer componente do *pipeline*, de forma a permitir a configuração inicial e reconfiguração dinâmica da aplicação a partir da monitoração do seu contexto de execução e parâmetros pré-definidos pelo usuário e/ou aplicação.

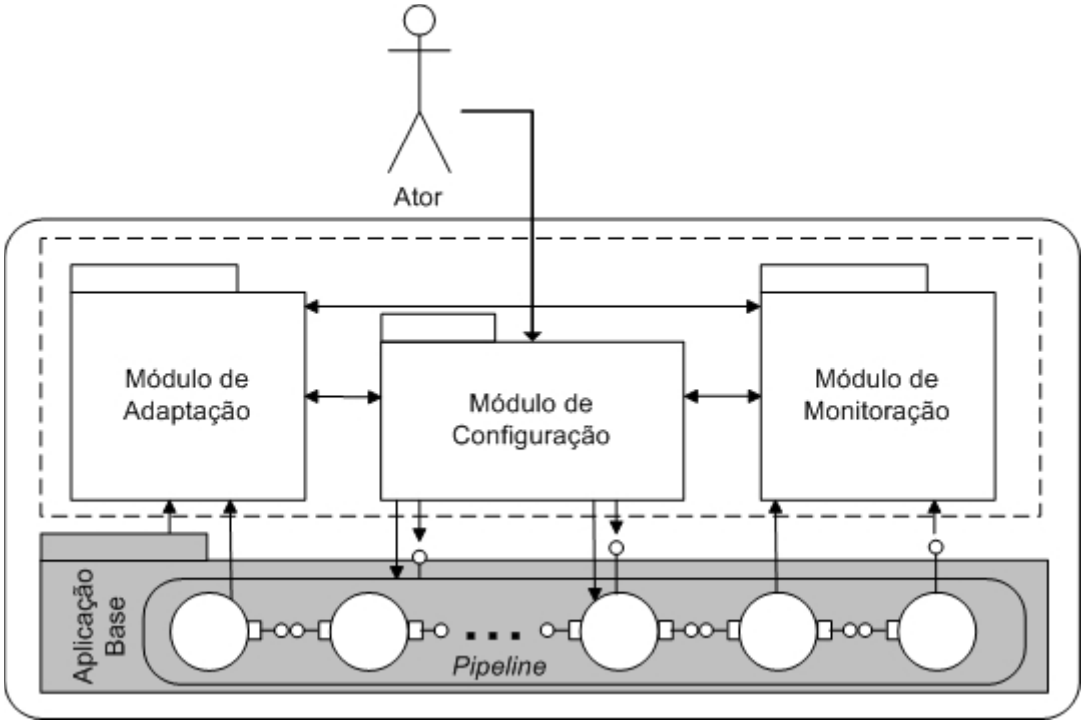

Figura 1: Estrutura do Adaptive Pipeline Aspect Pattern

5.1. Estrutura

O padrão contém 4 (quatro) módulos:

- **Aplicação Base:** módulo que representa os elementos com funcionalidades básicas de uma aplicação e estruturada de acordo com o padrão *Pipes and Filters*, mas sem nenhuma implementação de adaptabilidade.

- **Módulo de Configuração**: este módulo executa a configuração estática para cada *pipeline* durante a instanciação inicial de seus componentes de acordo com o estado atual do ambiente de execução e parâmetros passados pela aplicação ou usuário. Além disso, ele é responsável pela disponibilização de dados e implementação de reconfiguração dinâmicas (por exemplo, alterar ou substituir um componente do *pipeline*), ou

simplesmente políticas de adaptação, de acordo com mudanças reportadas pelo Módulo de Monitoração e decisões do Módulo de Adaptação. Isto significa que na situação de ter dois eventos iguais gerados pelo Módulo de Monitoração em momentos diferentes, podem implicar em comportamentos distintos de acordo com as políticas de adaptação fornecidas pelo Módulo de Configuração para o Módulo de Adaptação.

- **Módulo de Monitoração**: este módulo representa elementos genéricos que devem conter funcionalidades de monitoração no contexto de execução da aplicação-base, permitindo implementar, por exemplo, monitores de *pipelines* e/ou monitores do sistema onde a aplicação é executada. A partir dos dados passados pelo Módulo de Configuração, os monitores passam a realizar alguma monitoração, reportando informações que podem gerar eventos de adaptação. Dessa forma, os monitores em conjunto, através da geração de eventos, poderão indicar a ocorrência de violações de políticas para o Módulo de Adaptação.

- **Módulo de Adaptação**: Este módulo possibilita a implementação de requisitos de adaptabilidade de forma não intrusiva na aplicação-base. Dessa forma, esse módulo, em conjunto com os Módulos de Configuração e Monitoração, permite realizar mudanças na aplicação-base perante variações dinâmicas reportadas pelo Módulo de Monitoração ou mudanças de parâmetros no Módulo de Configuração. Mecanismos de adaptação possuem um conjunto de políticas de adaptação. Na camada proposta, uma política de adaptação pode ser entendida como um conjunto de estratégias e parâmetros que representam uma configuração que pode ser empregada na aplicação-base. Nesse caso, a regra de adaptação consiste em realizar ações corretivas baseadas nas condições verificadas e nos requisitos e configurações especificadas no Módulo de Configuração e que podem mudar de forma dinâmica. O Módulo de Adaptação controla a instanciação inicial e finalização dos componentes que representam um *pipeline* da aplicação-base. Além disso, o Módulo de Adaptação se comunica com o Módulo de Configuração de forma a obter os dados para adaptação de forma de dinâmica (políticas de adaptação) e delegar para este mesmo módulo a execução da reconfiguração adequada.

5.2. Dinâmica

Nesta seção descrevemos um cenário onde é realizado a configuração inicial de um *pipeline* e dos elementos responsáveis pela monitoração, a geração de um evento e a execução de uma configuração dinâmica através da consulta das políticas de adaptação. Ao invés de representar objetos, cada caixa do diagrama (Figura 2) de seqüência representa uma coleção de aspectos e/ou objetos, que corresponde a um determinado módulo do padrão.

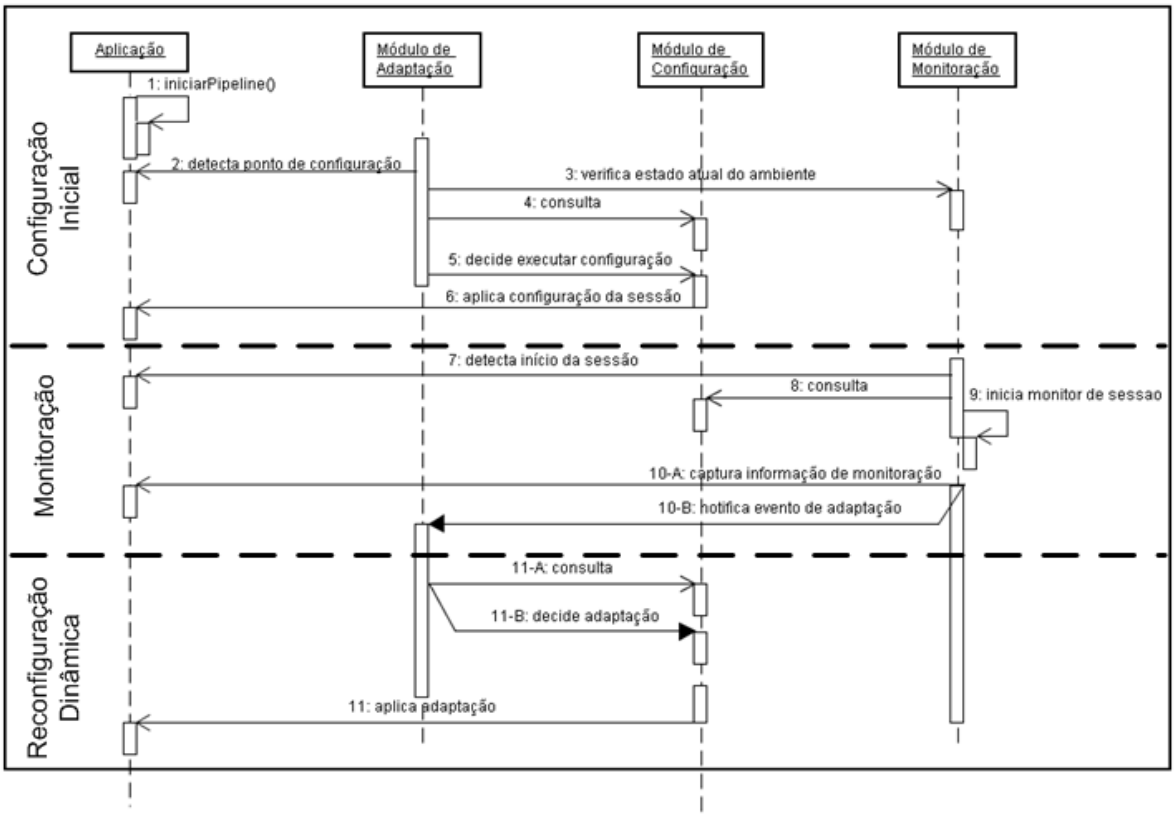

Figura 2: Dinâmica do Adaptive Pipeline Aspect Pattern

Abaixo são explicadas as três fases de colaboração do padrão *Adaptive Pipeline Aspect:*

- **Configuração Inicial:** (1) a aplicação inicia a execução e requisita a abertura de um *pipeline*; (2) o Módulo de Adaptação intercepta o(s) ponto(s) de execução (através da implementação do conceito de pontos de junção de POA) da criação de um *pipeline* onde uma configuração inicial deve ser realizada; (3, 4 e 5) o Módulo de Adaptação verifica o estado atual do ambiente de operação (por exemplo, carga atual do sistema ou número de *pipelines* instanciados) e realiza uma consulta ao Módulo de Configuração para decidir que configuração inicial deve ser realizada; (6) no Módulo de Configuração é implementada a configuração inicial na criação de um *pipeline* (por exemplo, estratégia de concorrência).
- **Monitoração:** (7, 8 e 9) o Módulo de Monitoração intercepta quando um processo de criação de *pipeline* é finalizado e inicia um processo de monitoração individual para a *pipeline* criado, realizando uma consulta ao Módulo de Configuração para verificar e decidir que política de monitoração que deve ser realizada. (10A) com a aplicação executando o Módulo de Monitoração captura constantemente informações de monitoração dos componentes do *pipeline* e decide se deve ou não gerar evento de adaptação
- **Reconfiguração Dinâmica:** (10B) quando um evento pré-configurado é gerado pelo Módulo de Monitoração, o Módulo de Adaptação é notificado; (11A) o Módulo de Adaptação consulta o Módulo de Configuração de forma a obter acesso a informações dinâmicas sobre como a adaptação deve ser executada; (11B e 11) o Módulo de Adaptação decide onde e como a configuração deve ser realizada e delega para o Módulo de Configuração mudar o comportamento da aplicação, por exemplo alterando uma propriedade de um componente ou substituindo por outro, de acordo com eventos de monitoração gerados e políticas de adaptação pré-configuradas.

6. Exemplos de Utilização

O padrão pode ser aplicado em:

- sistemas multimídia operando em redes sem reservas de recursos e garantia de QoS na camada de rede e subjacentes. Tal cenário exige que os sistemas ajustem, no nível da camada de aplicação, o seu funcionamento às condições variáveis que a rede comunicação de dados oferece.
- sistemas que utilizam protocolos de comunicação e demandam requisitos de alta disponibilidade e/ou flexibilidade: personalização no atendimento de requisições de cada usuário, atendimento de requisitos de tolerâncias a falha, reconfiguração dinâmica à carga imposta ao sistema e otimização de utilização de recursos disponíveis.

Além desses exemplos, o padrão também pode ser empregado em qualquer outro sistema de software que contenha funcionalidades representadas através do conceito de *Pipes and Filters* (*pipeline*) e exijam o processamento e transformações do fluxos de dados de forma adaptativa. Exemplos de funcionalidades incluem: (i) codificadores/decodificadores de áudio e/ou vídeo (ii) mecanismos de criptografia/decriptografia; e (iii) processadores/interpretadores (*parsers*) de mensagens de protocolos em sistemas web ou distribuídos (HTTP, SOAP, RMI, IIOP,etc.).

7. Implementação

Módulo de Adaptação: Este módulo deve ser implementado utilizando construções de programação orientada a aspectos, tais como *pointcuts* e *advices*, para definir e inteceptar pontos de execução (pontos de junção) da aplicação onde devem ser realizadas operações de configuração inicial ou reconfiguração dinâmica, que são delegadas para o Módulo de Configuração. Uma forma de facilitar a estruturação desse módulo é utilizar a expressão idiomática *Abstract Pointcut* [5], na qual usamos um aspecto abstrato para definir o comportamento comum (*advices*) dependendo de um *pointcut* abstrato. A partir desse aspecto abstrato, diferentes aspectos concretos podem ser implementados, os quais concretizam o *pointcut* abstrato. Isso permite a reutilização do aspecto abstrato em diversas aplicações adaptativas que compartilham interfaces para seu ciclo de vida e/ou conexão entre componentes de forma bem definida.

- **Módulo de Configuração**: Neste módulo devem estar presentes classes ou aspectos auxiliares para realizar alguma ação de (re)configuração na aplicação-base de acordo com pontos selecionados no módulo de adaptação e informações coletadas e eventos gerados por aspectos do módulo de monitoração. O isolamento dos outros módulos permite reusabilidade, já que vários aspectos do módulo de adaptação poderão usar essas classes ou aspectos que realizam uma determinada ação de adaptação (no caso, através do acesso direto a instância ou a uma interface de controle de um componente/objeto). Além disso, este módulo deve possuir elementos responsáveis pelo acesso a repositórios que provêem as informações sobre as políticas de adaptação e monitoração que podem ser realizadas pela camada de adaptação. Desta forma, mudanças dinâmicas na definição dessas políticas podem ser realizadas sem necessidade de expressar essas modificações diretamente no código da camada de adaptação, mas através de metadados. Pode ser usada uma abordagem simplificada para a definição dos metadados (arquivos no formato de texto e propriedades), utilização de XML, modelos para definição de políticas (por exemplo, *Policy Core Information Model*) e metamodelos (modelos que descrevem modelos) para configurações dos objetos [6]. Isto permitiria um comportamento ainda

mais dinâmico e flexível para a aplicação-base.

- **Módulo de Monitoração**: Este módulo deve ser desenvolvido de acordo com o tipo de observação do contexto que é exigido para realizar adaptações na aplicação-base. Dessa forma, diversos tipos de monitores podem ser implementados: monitores de sessões de *pipeline*, monitores individuais para cada componente, monitores para coletar estatísticas de utilização dos recursos do sistema computacional onde está sendo executada a aplicação-base, monitores para analisar o estado da rede ou canal de comunicação, etc. No caso de monitoração de elementos da própria aplicação ou de outro sistema de software, é recomendado utilizar o padrão *Observer*, particularmente a solução usando aspectos [7], de forma a permitir um modelo de monitoração mais flexível, não intrusivo e transparente para a aplicação. Adicionalmente, devem ser implementados elementos para realizar o processamento das informações de monitoração e verificar se deve ser gerado um evento de adaptação de acordo com condições pré-definidas.

8. Consequências

As principais vantagens do uso do padrão apresentado são:

+ separação do interesse de adaptabilidade de outros interesses, de forma a possibilitar reuso e facilidade de manutenção;
+ adicionar, remover e ativar e desativar facilmente as funcionalidades de adaptabilidade;
+ facilidade de manutenção das funcionalidades da aplicação-base de forma independente da funcionalidade de adaptação;
+ possibilidade de mudanças nas políticas de monitoração e adaptação sem afetar de forma significativa a estrutura geral da aplicação-base; e
+ acesso direto ao *pipeline* ou componentes através de pontos de junção (*join points*), sem necessidade da existência de interfaces.

Como desvantagens do uso do padrão temos:

- possibilidade de redução da eficiência quando comparado com soluções que se baseiam em arquiteturas monolíticas, já que é necessário a adição de vários elementos para implementar cada módulo, acarretando um aumento do tamanho do código e processamento para realizar adaptação;
- compilação do código-fonte da aplicação ficar mais lenta, devido o uso de orientada a aspectos que adiciona a operação de combinar os aspectos com o código original da aplicação; e
- necessidade de conhecimento do paradigma de programação orientada a aspectos.

9. Usos Conhecidos

O padrão arquitetural *Adaptive Pipeline Aspect* foi utilizado em [8] onde foi especificada a arquitetura de uma camada de adaptação para aplicações multimídia que utilizam a biblioteca *Java Stream Assembly* (JSA) [9] baseada em *Pipes and Filters*. Como forma de validação, a arquitetura foi instanciada com a linguagem AspectJ para uma aplicação servidora RTSP (*Real Time Streaming Protocol*) implementada em Java. Dois mecanismos de monitoração e de adaptação foram desenvolvidos com aspectos para ilustrar uma melhoria da qualidade de serviço na transmissão de vídeo através do uso de políticas de monitoração e adaptação.

Furfaro [10] apresenta o desenvolvimento de um mecanismo para sincronização de recepção de fluxos multimídia baseado em programação orientada a aspectos com AspecJ. No

trabalho é proposto um aspecto que fica responsável pela interceptação dos pacotes de fluxos multimídia e pode realizar operações de sincronização sobre os pacotes com objetivo de melhorar parâmetros de QoS (por exemplo, atraso e variação de atraso) na recepção da mídia. O aspecto *QoSFilter* apresentado no trabalho pode ser visto como uma implementação restrita do padrão aqui proposto.

O JBoss é um servidor de aplicação que implementa a especificação J2EE (*Java 2 Platform Enterprise Edition*) e permite a implantação de componentes dinamicamente em tempo de execução [11]. Como parte da arquitetura que permite essa funcionalidade, o JBoss possibilita que um conjunto de componentes interceptadores tratem chamadas tanto do lado do cliente, como no servidor de forma flexível. Tais interceptadores são implementados com programação orientada a aspectos e representam *pipelines* configuráveis que processam e alteram chamadas/mensagens a componentes do servidor, podendo implementar interesses tais como: segurança, transações, registro, monitoração e etc. Tal funcionalidade pode ser vista como uma implementação específica do padrão arquitetural *Adaptive Pipeline Aspect*.

10. Padrões Relacionados

- *Adaptive Pipeline* [3]: este padrão arquitetural propõe a separação dos filtros do *pipeline* e dos algoritmos que tais componentes utilizam para realizar a transformação de dados, permitindo que ambos sejam modificados independentemente e de acordo com a monitoração do seu ambiente de execução. Já o padrão apresentado permite essa separação, mas realiza-a de forma modular através de aspectos que encapsulam as funcionalidades de adaptação.
- AdapPE [12]: um padrão arquitetural que permite a estruturação de aplicações adaptativas utilizando orientação a aspectos. Enquanto o AdapPE é genérico, o *Adaptative Pipeline Aspect* se restringe a arquitetura de aplicações especificamente baseadas em *Pipes and Filters*.
- Auto-Adaptável [13]: permite o desenvolvimento de aplicações adaptativas, baseado em mudanças no seu ambiente de execução e utilizando um arquitetura modular e flexível. O padrão proposto aqui utiliza uma técnica mais avançada (POA) para separar os interesses de adaptação e além disso, prevê a monitoração também da aplicação que será adaptada.
- PLADS[14]: apresenta uma linguagem de padrões para sistemas adaptativos distribuídos. Além de utilizar outra técnica para separar os interesses de adaptação: reflexão (as adaptações devem ser realizadas através de um protocolo meta-objeto) [3], os padrões propostos em PLADS têm utilização mais genérica (não necessariamente, aplicações baseadas em *Pipes and Filters)* e consideram também adaptações em sistemas distribuídos.

11.Referências

[1] LYYTINEN, K. e YOO, Y. *Issues and challenges in ubiquitous computing*. Communications of the ACM, 2002, p. 45(12)62–65.

[2] KEPHART, J. O. e CHESS, D. M. *The Vision of Autonomic Computing*. IEEE Computer Magazine, jan., 2003.

[3] BUSCHMANN, F., MEUNIER, R., ROHNERT H., SOMMERLAD, P. e STAL M.. *A System of Patterns: Pattern-Oriented Software Architecture.* John Wiley& Sons, 1996.

[4] POSNAK, E.J., LAVENDER, R.G. e VIN, H.M. Adaptive pipeline: An object structural pattern for adaptive applications. *In Proceedings of the Third Pattern Languages of Programming Conference*, Illinois, Set., 1996.

[5] HANENBERG, S. e SCHMIDMEIER. Idioms for Building Software Frameworks in AspectJ. *In 2nd AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software(ACP4IS)*, Boston, mar., 2003.

[6] YODER, J. W., BALAGUER, F. e Ralph JOHNSON, R. *Architecture and Design of Adaptive Object-Models*. ACM SIGPLAN Notices, 2001. p. 36(12):50–60.

[7] HANNEMANN, J. e KICZALES, G.. Design pattern implementation in Java and AspectJ. *In Proceedings of the 17th ACM conference on Object-oriented programming, systems, languages, and applications*, pages 161–173. ACM Press, 2002.

[8] KULESZA, R., KULESZA, U. e BRESSAN, G. *Implementando uma Camada de Adaptação para Transmissão de Mídias usando Programação Orientada a Aspectos*. Simpósio Brasileiro de Sistemas Multimídia e Web (*to appear*). Brasil, 2006.

[9] *Java Stream Assembly API Programmer's* Guide. Sun Microsystems Inc. Disponível em: <http://www.jcp.org/en/jsr/detail?id=158>. Acesso em: 15 de setembro de 2006.

[10] FURFARO, A., NIGRO, L. e PUPO, F. *Multimedia synchronization based on aspect oriented programming*. Microprocessors and Microsystems, Elsevier, 2003, p. 47-56.

[11] FLEURY, M., REVERBEL, F. *The JBoss Extensible Server*. Middleware 2003 - ACM/IFIP/USENIX International Middleware Conference, vol. 2672, LNCS, pp. 344-373, Springer-Verlag, 2003.

[12] DANTAS, A. e BORBA, P., Adaptability Aspects An Architectural Pattern for Structuring Adaptive Applications with Aspects. *Proceedings of the 3th Latin American Conference on Pattern Languages of Programming* (SugarLoafPLoP'2003), ago., Brasil, 2003.

[13] CAMARGO, R. Y e QUEIROZ, C. A. *O Padrão Arquitetural Auto-adaptável*. Disponível em: <http://gsd.ime.usp.br/~kon/PLoP/2003/>. Acesso em: 15 de setembro de 2006.

[14] SILVA, F. J. S., KON, F., YODER, J. e JOHNSON, R. A Patttern Language for Adaptive Distributed Systems. *Proceedings of the 5th Latin American Conference on Pattern Languages of Programming* (SugarLoafPLoP'2005), pp. 19-48. Brasil. 2005.