UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

Relatórios Técnicos
do Departamento de Informática Aplicada
da UNIRIO
n°0013/2009

Programação orientada a aspecto no paradigma orientado a serviço: Uma análise sobre acoplamento tecnológico

Michel Silva
Flávia Santoro
Renata Araujo

Departamento de Informática Aplicada

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
Av. Pasteur, 458, Urca - CEP 22290-240
RIO DE JANEIRO – BRASIL

Relatórios Técnicos do DIA/UNIRIO, No. 00013/2009
Editor: Prof. Sean W. M. Siqueira Julho, 2009

Programação orientada a aspecto no paradigma orientado a serviço: Uma análise sobre acoplamento tecnológico

Michel Silva[1,2] , Flávia Santoro[1,2] , Renata Araujo[1,2]

[1]Depto de Informática Aplicada – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
[2] Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)

michel.silva@uniriotec.br, flavia.santoro@uniriotec.br, renata.araujo@uniriotec.br

Abstract. This paper presents a synthetic view in terms of aspect-oriented programming into a service-oriented computing (SOC) paradigm. This work proposes an understanding about the relationship between services and crosscutting concerns and demonstrates how to achieve better system modularization through the use of AOP in service-oriented applications development.

Keywords: aspect, AOP, service, SOA, SOC, ESB.

Resumo. Este artigo apresenta uma visão sintética da associação da programação orientada a aspectos no paradigma da computação orientada a serviço (SOC). Este trabalho propõe estabelecer um entendimento sobre dependências decorrentes da associação de tratamentos transversais aos serviços sistêmicos e demonstrar como alcançar uma melhor modularização do sistema através do uso de AOP no desenvolvimento de aplicações orientadas a serviços.

Palavras-chave: aspecto, AOP, serviço, SOA, SOC, ESB.

Sumário

1 Introdução

Com o advento da Web e o crescente investimento em aplicações de comércio eletrônico (*e-commerce*) e aplicações B2B, emerge a necessidade de cada vez mais desenvolver sistemas de informação aderentes às características de sistemas ubíquos, ou seja, sistemas distribuídos que utilizam uma grande rede para se comunicar. Porém, o desenvolvimento de sistemas é um processo custoso e demorado, conforme identificado pela Engenharia de Software, pode ser agravado por um exaustivo ciclo de manutenções inerente a erros inseridos em tempo de desenvolvimento. Tal fato está diretamente relacionado à inexistência de um nível de qualidade necessário e desejado ao processo de desenvolvimento de sistemas.

Atualmente, organizações utilizam-se de serviços para compartilhar informações referentes ao negócio, ou seja, informações de interesse comercial. Porém, essas informações podem estar distribuídas entre distintos sistemas e plataformas tecnológicas dentro da organização. Com isso, encontra-se nas organizações uma grande dificuldade em disponibilizar informações agrupadas e coerentes, pois sistemas construídos em momentos distintos e servindo ao mesmo propósito podem estar, também, em diferentes plataformas. Conforme [Pressman, 2005], entende-se que dificuldades inerentes ao desenvolvimento de sistemas podem ser agravadas pela falta de gestão sobre a evolução dos requisitos ao longo do tempo. Então, como disponibilizar um serviço coeso que agrega informações armazenadas em momentos distintos e que possua uma manutenibilidade adequada?

Este artigo tem como objetivo apresentar uma visão sintetizada sobre a questão do tratamento a comportamentos transversais em um paradigma da computação orientada a serviço. Buscando possibilitar esta realização, será exposta uma forma de atuar na ortogonalidade de serviços e suas respectivas integrações. Para descrever esta abordagem, será utilizado um modelo que procura demonstrar como alcançar uma melhor modularização de um sistema através do uso de AOP (*Aspect-Oriented Programming*) no desenvolvimento de aplicações orientadas a serviços.

Na seção 2 são apresentados conceitos provenientes da literatura sobre o paradigma de computação orientada a serviço. Em seguida, alguns problemas e propostas inerentes a ortogonalidade na computação orientada a serviço são descritas na seção 3. Conseguinte da seção 3, na seção 4 é estabelecido o foco em um modelo destinado a integrar serviços e, consequentemente, é apresentada uma visão da associação deste modelo com a programação orientada a aspectos para tratar comportamentos transversais oriundos das necessidades de composição dinâmica de serviços.

2 Orientação a Serviço

A Computação Orientada a Serviço (SOC) é considerada por [Papazoglou e Georgakopoulos, 2003] como o paradigma computacional que utiliza serviços como elementos fundamentais para o desenvolvimento de aplicações. Com isso, a SOC [Papazoglou *et al*. 2007] tem por objetivo promover a integração de componentes de aplicação formando, assim, uma rede de serviços flexíveis, desacoplados e distribuídos dentro do ambiente organizacional e plataformas tecnológicas existentes.

Para atingir os objetivos do paradigma da computação orientada a serviço uma arquitetura, nomeada como SOA (*Service-Oriented Architecture*), é adotada em nível de

infraestrutura tecnológica para definir, estruturar e implementar uma robusta camada de troca de mensagens. Esta infraestrutura fornece mecanismos que facilitam a integração de aplicações baseadas em serviços.

Serviços caracterizam-se por fornecer uma interface bem definida e estão associados à implementação de um determinado interesse do negócio, podendo esse ser um interesse funcional ou não-funcional. [Papazoglou e Heuvel, 2006] consideram que o desenvolvimento de serviços está diretamente associado ao processo de negócio. Logo se faz necessário derivar o modelo de processo de negócio para efetivamente construir os serviços da organização considerando, assim, a reusabilidade inerente da independência de aplicação e plataformas que irão executá-los.

Porém, para garantir a reusabilidade do serviço é necessário identificar a granularidade que o mesmo apresenta no escopo do negócio. Por isso, é importante estabelecer a devida composição e relacionamento entre eles, sem afetar características como sua forte coesão e seu fraco acoplamento. Pode-se, então, denominar serviços, de acordo com sua estrutura, como serviços componentes e serviços compostos [Papazoglou e Heuvel, 2006], sendo serviços compostos aqueles formados por um conjunto de serviços componentes. Portanto, considera-se serviços componentes como serviços de granularidade fina e serviços compostos como serviços de granularidade mais grossa, ou seja, são, respectivamente, serviços mais detalhados e serviços de mais alto nível.

Conforme [Huhns e Singh, 2005], serviços são disponibilizados em um componente de registro de serviços, sendo oferecidos por provedores de serviços e utilizados por consumidores, ou seja, clientes destes serviços. A utilização de serviço mais comumente difundida é o serviço Web. [Papazoglou e Heuvel, 2007], [Huhns e Singh, 2005] e [Binildas, 2008] concordam que provedores e consumidores são fracamente acoplados no contexto da computação orientada a serviço. O serviço Web fornece a interoperabilidade entre aplicações através de troca de mensagens. A comunicação é realizada pela Web por meio de padrões preestabelecidos, o que inclui o SOAP (*Simple Object Access Protocol*) para transmissão de dados e o WSDL (*Web Services Description Language*) para a definição dos serviços Web.

Contudo, a definição de uma arquitetura de serviços implementada através de ativos de infraestrutura se faz necessária para fortalecer uma estrutura homogênea de integração de serviços heterogêneos e distribuídos. Então, SOA utiliza-se de uma tecnologia de *middleware* denominada ESB (*Enterprise Service Bus)*.

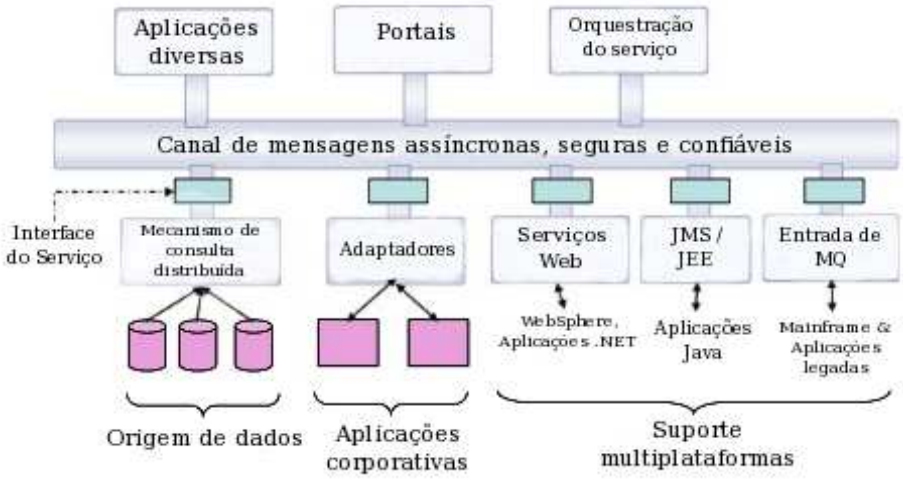

Fig. 1: O Enterprise Service Bus conectando diversas aplicações

2.1 O Enterprise Service Bus

Para [Papazoglou *et al*. 2007], o ESB visa garantir um canal único de roteamento de mensagens e consequente integração dos serviços com um baixo acoplamento e dispondo de um facilitado gerenciamento dos serviços. Por isso, [Papazoglou e Heuvel, 2007] define o ESB como uma plataforma de integração apoiada por uma variedade de padrões de comunicação em diversos protocolos de transporte e adiciona habilidades de entrega para aplicações SOA. Portanto, o ESB facilita a integração de distintas aplicações sistêmicas e plataformas tecnológicas. O ESB é defino por [Binildas, 2008] como uma plataforma que contém uma coleção de serviços de middleware, a qual fornece habilidades de integração, como o roteamento e transformação de mensagens, através de conectores inteligentes.

Em [Binildas, 2008] é apresentada uma relação de funcionalidades e características para facilitar o entendimento sobre arquitetura do ESB. Portanto, a relação é estabelecida da seguinte forma:

- Endereçamento e roteamento;
- Estilo síncrono e assíncrono;
- Ligação entre múltiplas formas de transporte e protocolos;
- Orquestração de processo de negócio;
- Processamento de eventos;
- Adaptadores para múltiplas plataformas;
- Integração com ferramentas de projeto, implementação e implantação;
- Características de QOS como transação, segurança e persistência;
- Auditoria, depuração de eventos e medição;
- Gerenciamento e monitoramento;

No contexto do ESB, os serviços dispõem de uma descrição associada, como o WSDL do serviço Web, e esta descrição fica exposta no ESB. Com isso, os serviços provedores não expõem detalhes de sua implementação e ficam disponíveis para facilmente serem localizados por serviços consumidores. Contudo, uma questão emerge: como tornar serviços, que podem até estar dispostos em plataformas e tecnologias distintas, interoperáveis em um ESB?

Para responder a questão acima direcionada, [Binildas, 2008] apresenta o JBI (*Java Business Integration*)[1]. O JBI é um *framework* de colaboração, o qual provê interfaces padronizadas para a integração plugável de componentes e protocolos no ESB, permitindo assim a montagem de um *framework* SOI (*Service Oriented Integration*)[2]. [Binildas, 2008] diz que o JBI tem como objetivo atender princípios de SOA e SOI, portanto, para este fim, o JBI é construído ao redor do WSDL. Com isso, aplicações podem ser disponibilizadas e integradas dentro do ambiente do JBI utilizando-se de um modelo basea-

[1] Conforme sua nomenclatura, percebe-se que este componente é escrito em Java. No contexto do JEE, a especificação do JBI é estabelecida na JSR-208.

[2] SOI é definido por [Binildas, 2008] como soluções de projeto para problemas recorrentes, ou seja, um dos design patterns da SOA. Diz ainda que é o melhor caminho para o reuso de serviços.

do no WSDL. Para fornecer um melhor entendimento sobre a arquitetura do JBI, [Binildas, 2008] apresenta o seguinte diagrama:

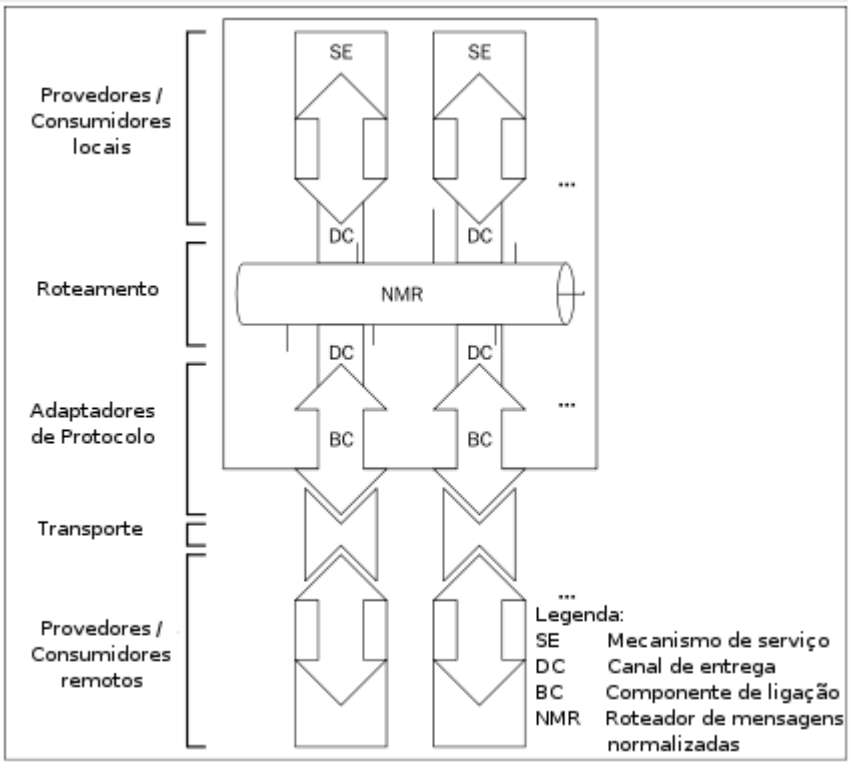

Fig. 2: A arquitetura do JBI

Em [Binildas, 2008], os elementos da arquitetura são contextualizados da seguinte forma:

- Ambiente do JBI – é simplesmente uma máquina virtual Java (JVM) onde podem ser disponibilizados objetos de integração.
- Mecanismo de serviço (SE) – são serviços provedores ou consumidores disponibilizados localmente no ambiente JBI, sendo responsáveis pela lógica de negócio como, por exemplo, a transformação.
- Componente de ligação (BC) – provê suporte ao protocolo de comunicação, sendo o responsável pela integração dos provedores e consumidores de serviços locais com os respectivos de serviços remotos.
- Roteador de mensagens normalizadas (NMR) – é o ponto central da arquitetura JBI. Este é o componente responsável pelo roteamento de mensagens normalizadas[3] entre a origem e o destino.
- Canal de entrega (DC) – conecta a mensagem da origem e o destino, ou seja, responsável pela lógica de endereçamento de um ESB.

[3] Mensagem normalizada consiste em uma mensagem composta pela própria mensagem em formato XML e a mensagem de metadados, o qual determina o contexto dos dados da mensagem. O NMR só endereça mensagem estabelecidas neste formato.

Tendo visto o grande potencial de integração de serviços corporativos proporcionado pela arquitetura do ESB, veremos, então, nas seguintes seções, onde esta arquitetura não consegue alcançar a fim de estabelecer uma completa solução de integração entre serviços e interesses transversais envolvidos durante todo o fluxo de execução do processo organizacional, ou seja, o processo que é iniciado em uma camada de aplicativo cliente, passando posteriormente para a camada de serviços integrados e finalizando o fluxo processual através do seu retorno, positivo ou não, para o respectivo cliente solicitante.

3 Ortogonalidade no paradigma orientado a serviço

Pode-se observar, então, que serviços fornecem uma implementação estabelecida para tratar interesses inerentes da aplicação e esses serviços podem compor outros serviços, ou seja, há uma forte relação entre serviços e estes serviços podem implementar interesses que reusam alguns dos interesses preexistentes no contexto sistêmico da organização.

Para [Laddad, 2003], esses interesses são especificados no momento que é feito o levantamento de requisitos, onde ficam estabelecidas as necessidades e os objetivos do sistema de informação. Com isso, pode-se dizer que um serviço bem definido é todo aquele que atende a um determinado interesse sistêmico da organização e pode ser considerado um elemento modular, ou seja, um elemento autônomo com características de possuir um baixo acoplamento e alta coesão.

Modularizar, ou separar interesses, implica em garantir que problemas como código emaranhado[4] e código espalhado[5] não aconteçam. A modularização proporciona inúmeras vantagens para o processo de desenvolvimento de software, como, por exemplo, rastreabilidade, produtividade, reusabilidade, manutenibilidade e um nível elevado de qualidade.

A programação orientada a aspecto, segundo [Laddad, 2003] e [Kiczales e Mezini, 2005], fornece mecanismos de composição modular facilitando, assim, a separação dos interesses sistêmicos, diferentemente das linguagens de componentes as quais direcionam a implementação dos interesses em pontos distintos dentro do código-fonte da aplicação. Por isso, o paradigma orientado a aspecto facilita, na atividade de análise de requisitos no contexto do desenvolvimento de sistemas, pensar nos interesses sistêmicos de forma transversal. Isso possibilita que haja uma melhor divisão de tarefas e, consequentemente, um rico levantamento de requisitos. Portanto, essa separação é refletida na qualidade do produto.

Com o auxílio desses mecanismos proporcionados pela AOP associados ao paradigma da SOC favorece-se a efetiva gestão dos serviços disponibilizados em uma organização. Portanto, muitos trabalhos propõem a integração do paradigma orientado a serviço com o paradigma orientado a aspecto, como [Mendonça *et al*, 2007], [Hmida *et al*, 2005] e [Cibrán *et al*, 2007]. Todos os trabalhos anteriormente mencionados utilizam serviços Web como elemento principal de sua abordagem em atingir a integração do paradigma orientado a serviço com o paradigma orientado a aspecto.

Em [Mendonça *et al*, 2007], uma linguagem de aspectos é proposta para atuar na combinação de interesses transversais em serviços Web. Essa linguagem é nomeada

---

4 Acontece quando o módulo é implementado para tratar, simultaneamente, múltiplos interesses.

5 Acontece quando uma simples necessidade é implementada em diversos módulos.

WSAL (*Web Service Aspect Language*) e se propõe a implementar os conceitos e componentes da AOP para o contexto da COS. Portanto, considera-se como *join points* atributos como URIs referentes a *namespaces*, partes da mensagem que é trocada entre o provedor e o consumidor do serviço, as operações que o serviço disponibiliza, a URL de localização do serviço e a URL de localização do cliente. Esta proposta atua em uma camada de rede, onde uma aplicação cliente solicita um serviço em um dado provedor e esta comunicação é interceptada através de um componente proxy. Este proxy exerce o papel do combinador de aspectos, o qual tem seu comportamento definido através de um artefato XML que define em quais *pointcuts*[6] o *serviço aspectual*[7] atuará. Esta interceptação pode ser tanto na solicitação quanto na resposta proveniente do provedor do serviço. Após a atuação do *serviço aspectual*, o combinador aspectual prossegue com a troca de mensagens entre as partes.

Na abordagem proposta por [Hmida *et al*, 2005] tenta-se chegar ao mesmo objetivo de [Mendonça *et al*, 2007] em estabelecer uma abordagem que seja independente de tecnologia de implementação. Com isso, [Hmida *et al*, 2005] fundamenta sua abordagem arquitetural usando três componentes básicos: um XML Schema, onde descreve como o aspecto deve ser especificado e declarado, um módulo gerador de XQuery, o qual recebe como entrada um arquivo XML contendo a especificação do aspecto e, com isso, gera um script XQuery para invocar um serviço Web, e um mecanismo de XQuery, onde o script gerado é, então, executado. Portanto, o mecanismo de XQuery será o responsável por executar a combinação aspectual através do uso do script gerado.

Contudo, [Cibrán *et al*, 2007] segue uma linha diferente das abordagens anteriormente propostas. Nesta abordagem, a preocupação é focada no contexto das aplicações cliente. Então, propõem-se uma nova camada chamada WSML (*Web Services Management Layer*) para controlar a integração e configuração de serviços Web nas aplicações cliente. Esta camada situa-se entre o consumidor e o provedor do serviço, ou seja, entre a aplicação cliente e o servidor. O cliente solicita um serviço a esta camada através da chamada a um serviço Web (*Service Type*). Na WSML os serviços são combinados e invocados através de aspectos que associados a alguns atributos coletados por um monitor de serviços classificam dinamicamente qual o serviço mais adequado para responder a uma solicitação.

4 Um modelo para a convergência entre aspectos e serviços

Na seção anterior foram apresentadas algumas abordagens propostas para a integração entre aplicações SOA baseadas em serviços Web e linguagens da AOP. No entanto, o propósito da COS está muito além do serviço Web. Com isso, emerge a necessidade de integrar aplicações existentes, legadas, e aplicações que estão em processo de desenvolvimento. Portanto, esta seção apresenta uma proposta de modelo que viabilize a possibilidade de convergência das aplicações existentes em plataformas tecnológicas da organização e aplicações provenientes de novas demandas de desenvolvimento.

Com isso, o uso de uma tecnologia de integração torna-se fator primordial para atingir o ápice da integração de serviços perseguido pela computação orientada a servi-

---

[6] Componente da AOP que define um conjunto de *join points*.

[7] Implementa um interesse transversal em forma de um serviço Web, ou seja, ele exerce a função do componente *advice*.

ço. Então como visto anteriormente, o ESB atende a este requisito e será adotado como elemento base da estrutura deste modelo.

Porém, é necessário exercitar a separação dos interesses considerando o domínio do negócio em que o sistema está envolvido visando, assim, a modularização sistêmica e a devida identificação dos serviços existentes no ambiente corporativo em tempo de análise. [Pressman, 2005] explica que a engenharia de requisitos constrói a ponte para o projeto e construção do sistema em face da necessidade de garantir que o sistema correto seja construído.

A tendência de unificar sistemas legados e aplicativos novos torna mais complexo determinar as conexões inerentes aos seus requisitos. Então, a gestão dos requisitos, presente na engenharia de requisitos e definida por [Pressman,2005] como um conjunto de atividades que ajudam a equipe de projeto a identificar, controlar e rastrear requisitos e modificações de requisitos em qualquer momento da evolução do sistema, será fator importante no contexto deste trabalho. À medida que esta rastreabilidade é identificada, projetistas e desenvolvedores podem visualizar qual o relacionamento existente entre os interesses que compõem o sistema.

Como visto anteriormente, [Papazoglou e Heuvel, 2006] consideram que o desenvolvimento de serviços é proveniente do processo de negócio. Então, [Azevedo *et al*, 2009] analisam que propostas apresentadas na literatura não descrevem em um nível de detalhes adequado a atividade de identificação de serviços a partir da modelagem de processos e, assim, apresenta uma abordagem de identificação de serviço a partir de modelos de processo de negócio. Porém, é muito difícil obter de qualquer organização o seu processo de negócio completamente modelado. Com isso, se faz necessário olhar para requisitos inerentes aos sistemas legados e pré-existentes com o intuito de complementar esta modelagem.

Então, para o caso da integração entre sistemas legados e sistemas novos faz-se importante analisar a percepção singular dos requisitos, fornecida pela modularização, em conjunto com a capacidade de rastreamento, a qual evidencia suas reais dependências, ou seja, seus relacionamentos. Com isso, a identificação de serviços existentes e os que venham a surgir com seus respectivos relacionamentos é facilitada por esta atividade de análise modular de requisitos, de acordo com o conceito estabelecido pela AOP sobre a separação dos interesses.

Portanto, após a identificação dos serviços originários da composição dos requisitos e tendo seus contratos bem estabelecidos, deve-se disponibilizá-los dentro da arquitetura do ESB estabelecendo, assim, uma gestão centralizada sobre o fluxo da execução dos processos organizacionais baseados em serviços. Então, a configuração dos serviços no ESB, como visto na seção 2, será realizada através de artifícios proporcionados pelo JBI. Com isso, percebe-se que o ESB fornece mecanismos que possibilitam a materialização da proposta de [Mendonça *et al*, 2007], sendo, assim, caracterizado como funcionalidade inerente da plataforma de integração.

Contudo, o ESB não soluciona todos os problemas. Interesses transversais provenientes do contexto de aplicações remotas provedoras e/ou consumidoras de serviços não serão totalmente atendidos por um ESB, pois compreende ambientes distintos de disponibilização de aplicativos de serviços. Para isso, necessita-se elaborar uma forma universal de interpretação de interesses transversais e que seja extensível para ser utilizada em aplicações desenvolvidas por linguagens de programação distintas. Então, entende-se que um arquivo XML prontamente atende a este requisito apontado.

Todavia, este XML deve possuir uma interface bem especificada, possibilitando, no contexto da linguagem de origem do aplicativo remoto, o desenvolvimento de interpretadores da AOP, sendo, esse, objeto essencial na devidamente interpretação deste respectivo artefato. Esse artefato será utilizado com a finalidade de acionar serviços, disponibilizados no contexto do ESB, que conterão a implementação para atender a transversalidade dos aplicativos remotos. Portanto, a abordagem proposta por [Cibrán *et al*, 2007] é amplamente atendida com a utilização de um ESB.

5 Conclusão

Este documento apresentou conceitos relacionados ao paradigma da computação orientada a serviço, descreveu como o ESB atende aos objetivos da SOA relacionando características inerentes a esta arquitetura e explicando seus componentes agregados. Descrevendo alguns conceitos oriundos da AOP e descrevendo algumas abordagens proposta sobre a integração do paradigma orientado a aspecto no orientado a serviço foi possível estabelecer um entendimento sobre a necessidade da atuação da AOP no paradigma orientação a serviços.

Da apresentação das abordagens propostas pode-se concluir que todos são atendidos através do correto uso de uma solução de ESB. Porém, o ESB não é uma solução para todos os problemas e foi visto que ainda não foi encontrada uma forma padrão para atender os interesses transversais atuantes no contexto das aplicações executadas fora do ambiente do ESB, dentre as quais podem estar desenvolvidas em linguagens de programação distintas.

Descrevendo a necessidade de estabelecer um padrão para a separação de interesses nas aplicações externas ao ESB é, então, vislumbrado o direcionamento de uma proposta para a criação de uma especificação padronizada, conforme as existentes no contexto do desenvolvimento de aplicações Java, ou seja, nos moldes da JSR (*Java Specification Requests*). Portanto, através desta especificação seria possível elaborar componentes, desenvolvidos em diferentes linguagens de programação, que efetivamente executariam este contrato.

Apresentou-se um modelo no qual utiliza-se de gestão no levantamento de requisitos baseado no conceito de separação de interesses, o qual garante a modularização e, consequentemente, agrega melhor legibilidade, reusabilidade, manutenibilidade e um nível elevado de qualidade, o qual é perseguido para o desenvolvimento e pretende-se que perdure após a efetiva implantação dos serviços, ou seja, no processo de evolução dos serviços organizacionais. Com o uso correto das ferramentas disponibilizadas pelo ESB é possível integrar, de forma segura e confiável, aplicações dispostas em ambientes heterogêneos e construídas para atender as diferentes necessidades da organização visando assim atender a um único propósito, o de alavancar a inteligência do negócio, ou seja, torná-lo promissor e competitivo.

Então, um ponto interessante para pesquisa seria utilizar a informação obtida da rastreabilidade para construir um sistema baseado na composição de interesses, ou seja, a composição aspectual do código sistêmico baseada no relacionamento entre os requisitos levantados. Porém, é necessário identificar mais características transversais relacionadas aos requisitos e, baseado nesta identificação, estabelecer uma forma de estruturar o serviço através do relacionamento obtido entre os próprios requisitos relacionados no contexto da rastreabilidade, a qual é devidamente instruída pela engenharia de requisitos.

6 Referências Bibliográficas

Azevedo, L.G.; Baiao, F. A.; Santoro, F. M.; Souza, J.; Revoredo, K.; Pereira, V. **Identificação de Serviços a partir da Modelagem de Processos de Negócio.** V Simpósio Brasileiro de Sistemas de Informação, Brasília, v. 1. p. 133-144, 2009.

Binildas, C. A. **Service Oriented Java Business Integration: Enterprise Service Bus integration solutions for Java developers.** Birmingham, UK: Packt Publishing Ltd., 2008.

Cibrán, M. A.; Verheecke, B.; Vanderperren, W.; Suvée, D.; Jonckers, V. **Aspect-oriented Programming for Dynamic Web Service Selection, Integration and Management.** World Wide Web Journal, v.10, n.3, pp. 211-242, 2007.

Hmida, M. M. B.; Tomaz, R. F.; Monfort, V. **Applying AOP Concepts to Increase Web Services Flexibility.** Proceedings of the International Conference on Next Generation Web Services Practices, 2005.

Huhns, M. N.; Singh, M. P. **Service-Oriented Computing: Key Concepts and Principles.** IEEE Internet Computing, vol. 9, no. 1, pp. 75-81, Jan./Feb. 2005.

Kiczales, G.; Mezini, M. **Aspect-Oriented Programming and Modular Reasoning.** Proceedings of the 27th international conference on Software engineering, pp. 49-58, 2005.

Laddad, R. **AspectJ in Action: Practical Aspect-Oriented Programming.** Greenwich, CT: Manning Publications Co., 2003.

Lesiecki, N.; Gradecki, J. **Mastering Aspectj: aspect-oriented programming in java.** Indianapolis, IN: Wiley Publishing, Inc., 2003.

Mendonça, N. C.; Silva, C. F. **Aspectual Services: Unifying Service- and Aspect-Oriented Software Development.** Proceedings of the International Conference on Next Generation Web Services Practices, 2005.

Mendonça, N. C.; Silva, C. F.; Maia, I. G.; Cordeiro, T. **Uma Linguagem para Especificação e Combinação Dinâmica de Aspectos em Aplicações Orientadas a Serviços.** XI Simpósio Brasileiro de Linguagens de Programação, 2007.

Papazoglou, M. P.; Georgakopoulos, D. **Introduction.** Communications of the ACM, v.46, n.10, pp. 24-28, 2003.

Papazoglou, M. P.; Heuvel, W.-J. V. D. **Service-oriented design and development methodology.** Int. J. Web Engineering and Technology, Vol. 2, No. 4, pp. ,412-442 2006.

Papazoglou, M. P.; Heuvel, W.-J. V. D. **Service oriented architectures: approaches, technologies and research issues.** The VLDB Journal,16:389–415, 2007.

Papazoglou, M. P.; Traverso, P.; Dutsdar, S.; Leymann, F. **Service-Oriented Computing: State of the Art and Research Challenges.** IEEE Computer Society, vol. 40, issue 11, pp. 38-45, 2007.

Pressman, R. S. **Software Engineering: A Practitioner's Approach.** 6 ed. New York, USA: McGraw-Hill, 2005.