Infra-estrutura de Serviços na Biblioteca de Componentes Brechó

Anderson Marinho[1], Cláudia Werner[1], Leonardo Murta[2]

[1] PESC/COPPE – Universidade Federal do Rio de Janeiro
Rio de Janeiro – RJ – Brazil

[2] Instituto de Computação – Universidade Federal Fluminense
Niterói – RJ – Brazil

{andymarinho, werner}@cos.ufrj.br, leomurta@ic.uff.br

Abstract. *A Software Component Library is a tool that supports component based development (CBD). Generally, most of the existing repositories only provide components physically, not considering another important way of providing them: as services. Therefore, this paper presents an infrastructure for providing component services in a software component library. This infrastructure deals with access control, utilization monitoring, components to services conversion, and composite services generation issues.*

1. Introdução

Reutilização de software é a disciplina responsável por criar sistemas de software a partir de software existente (Krueger 1992). Um dos paradigmas utilizados na reutilização é o Desenvolvimento Baseado em Componentes (DBC), que visa o desenvolvimento de componentes de software interoperáveis para serem reutilizados na construção de novos sistemas, garantindo, assim, melhores índices de qualidade e produtividade (Szyperski 1997). No DBC, uma ferramenta essencial é a biblioteca de componentes, que consiste em um local de publicação, armazenamento, busca e recuperação de componentes (Sametinger 2001).

Normalmente, a forma de disponibilização de componentes em uma biblioteca é por meio do fornecimento físico dos seus artefatos. Isso significa que, para se utilizar um componente, devem-se baixar seus arquivos e implantá-los em uma plataforma compatível. Contudo, em alguns cenários esta forma de utilização apresenta algumas desvantagens como, por exemplo, a baixa interoperabilidade, que torna difícil a reutilização do componente em diferentes plataformas. Além disso, diversas organizações estão adotando a terceirização, transferindo para terceiros suas infra-estruturas de desenvolvimento e produção. Com isso, uma alternativa para superar estes problemas é a disponibilização de componentes na forma de serviços.

Serviços solucionam o problema de interoperabilidade e terceirização, uma vez que estes não precisam ser implantados em plataformas específicas para serem utilizados, mas sim requisitados e executados em provedores remotos por meio de linguagens e protocolos padrões, comuns em qualquer ambiente de tecnologia de informação (Papazoglou 2003). Além disso, a tecnologia de serviços é considerada um importante passo para a computação distribuída (Papazoglou 2003), já que serviços podem ser ofere-

cidos por diferentes empresas através da Internet e, com isso, novas soluções ou aplicações podem ser construídas por meio da colaboração das mesmas. Mais ainda, a computação sob demanda que vem avançando tem como base a utilização de serviços. Futuramente, anseia-se que o processamento não será feito mais localmente, mas sim comprado na forma de serviços oferecidos na Web, funcionando de maneira similar a um serviço de eletricidade ou de comunicação (Smith et al. 2006).

O objetivo deste trabalho é apresentar uma infra-estrutura para aplicação de serviços em uma biblioteca de componentes, viabilizando o fornecimento de componentes não apenas fisicamente, mas também a partir de seus serviços. Esta infra-estrutura foi implementada na biblioteca Brechó e trata de questões referentes a controle de acesso e registro de utilização de serviços, além de mecanismos que auxiliem a publicação e o provimento de serviços. O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta a infra-estrutura de serviços implementada na Brechó e a Seção 3 conclui o artigo, apresentando alguns trabalhos relacionados, limitações e trabalhos futuros.

2. Serviços na Biblioteca de Componentes Brechó

Esta seção apresenta a infra-estrutura de serviços implementada na biblioteca de componentes Brechó. A biblioteca Brechó consiste de um sistema de informação web que visa proporcionar mecanismos de documentação, armazenamento, publicação, pesquisa e recuperação de componentes de software (Werner et al. 2007). Com isso, para a melhor compreensão dessa infra-estrutura, alguns detalhes sobre a implementação da Brechó serão discutidos ao longo desta seção.

Uma das preocupações na definição dessa infra-estrutura é que serviços sejam disponibilizados na Brechó de maneira análoga a componentes físicos, ou seja, que possam ter mecanismos de controle de acesso e registro de utilização. Estes mecanismos são essenciais para que a biblioteca não participe apenas na etapa de descoberta, mas também na etapa de utilização dos serviços. A partir disso, a biblioteca pode usufruir de diversas outras funcionalidades. Por exemplo, serviços podem ser tarifados facilmente se estiverem sob controle de acesso, e isso é um ponto crucial em especial para uma biblioteca comercial, já que disponibilizar serviços envolve custos. Um outro exemplo, a partir do registro de utilização dos serviços, a biblioteca pode fornecer informações importantes para os seus usuários. Por exemplo, um produtor pode ajustar o preço dos seus serviços de acordo com a sua freqüência de uso. Já um consumidor, pode usar essas informações como parâmetro de confiabilidade sobre os serviços que irá utilizar.

Nesta infra-estrutura, foram definidas três estratégias de publicação de serviços: *serviços externos*, para serviços já existentes; *serviços internos simples*, para serviços gerados a partir de componentes previamente publicados na Brechó; *serviços internos compostos*, para serviços construídos pela composição de serviços externos e internos simples ou de outros compostos. A primeira estratégia assume que os produtores já possuem serviços hospedados em um servidor e querem apenas publicá-los na Brechó. A segunda e terceira estratégias assumem que os produtores não possuem recursos para prover serviços e necessitam que a Brechó realize esse papel. Na segunda estratégia, porém, os serviços são gerados a partir de componentes previamente publicados, ao passo que, na terceira, os produtores constroem especificações BPEL (Business Process Execution Language) (OASIS 2007) dos serviços compostos e publicam na Brechó para que esta realize a geração, hospedagem e orquestração dos mesmos. BPEL é uma das

linguagens de composição de serviços mais conhecidas e vem sendo amplamente utilizada no mercado. BPEL possui diversas características que facilitam a construção de processos de negócios, tais como sintaxe robusta, possibilidade de definição de processos em níveis abstratos e concretos, dentre outras.

De maneira geral, cada estratégia apresenta algumas dificuldades na sua execução. Com relação aos serviços externos, a dificuldade encontra-se na inserção dos mecanismos de controle de acesso e de registro de utilização, uma vez que os mesmos são implantados fora da Brechó. Já os serviços internos simples, o principal problema está relacionado com a geração de serviços a partir de componentes. Esta geração é complexa, pois um componente pode estar codificado em qualquer linguagem de programação, e cada linguagem possui uma forma distinta de geração de serviços. Quanto aos serviços compostos, a principal dificuldade se encontra na implementação de uma máquina para executar esses serviços. Esta máquina deve ser capaz de instanciar um serviço composto a partir de uma especificação BPEL e orquestrá-lo.

Na Figura 1.a é apresentado a arquitetura da biblioteca Brechó estendida com a infra-estrutura de serviços. O objetivo principal desta arquitetura é manter a Brechó apenas com funcionalidades básicas de gerenciamento de componentes e serviços. As funcionalidades mais complexas de publicação e geração de serviços são transferidas para módulos específicos. Estes módulos podem ser classificados em três tipos: Módulos Específicos de Linguagem, que geram serviços internos simples a partir de componentes codificados a partir de uma linguagem de programação; Módulo Legado, que adapta serviços externos para trabalharem de acordo com o mecanismo do controle de acesso e registro de utilização da Brechó; e Módulo orquestrador, que produz serviços internos compostos a partir de descritores BPEL.

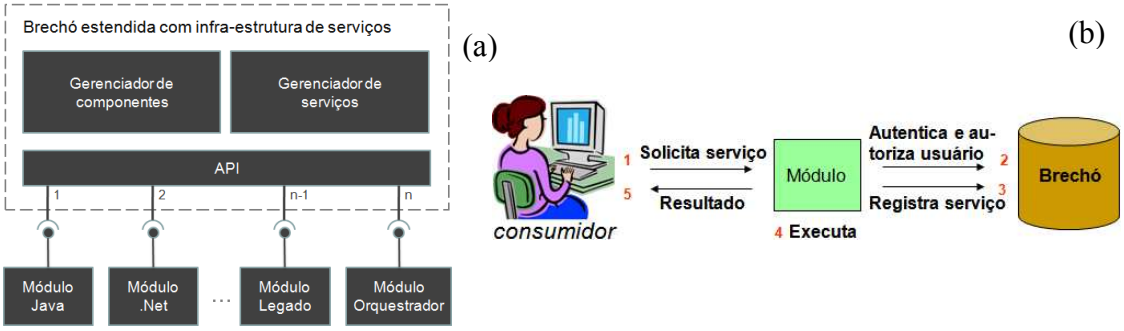

Figura 1 – Arquitetura da Brechó (a) e processo de utilização de serviços (b)

A comunicação entre módulos e a Brechó é feita por meio de uma API de serviços definida pela Brechó. Esta API define, em alto nível, as operações que os módulos devem implementar. O processo de utilização de serviços, onde módulo e Brechó se comunicam, é apresentado na Figura 1.b. Em um primeiro passo, o consumidor solicita o serviço, passando informações de autenticação da Brechó, assim como parâmetros do próprio serviço. Essa solicitação do serviço vai diretamente para o módulo, pois é onde o serviço está hospedado. Logo após, o módulo entra em contato com a Brechó para realizar a autenticação e verificar se o usuário é autorizado a usar o serviço ou não (e.g., verifica se o usuário possui créditos suficiente). Em caso positivo, a requisição ao serviço é registrada na Brechó e, posteriormente, o serviço é executado no módulo. Por fim, o resultado do serviço é retornado para o usuário.

As próximas seções abordam os três tipos de módulos de serviços, descrevendo suas principais funcionalidades.

2.1 Módulos Específicos de Linguagem

Esses módulos são responsáveis pela geração de serviços a partir de componentes de software. Em princípio, cada módulo tem a responsabilidade de gerar serviços em uma linguagem de programação específica. Isso é necessário, pois não existe uma solução genérica de construção de serviços que funcione em qualquer linguagem. A partir disso, a biblioteca, de acordo com a sua necessidade, adiciona ou remove módulos específicos de linguagem. Por exemplo, se a Brechó trabalha apenas com componentes Java e .Net, esta precisa adicionar um módulo para cada linguagem (Figura 1.a).

A arquitetura de um módulo específico de linguagem pode variar de linguagem a linguagem. Por esse motivo, escolhemos um módulo específico em Java para apresentar a arquitetura. A arquitetura do módulo Java é composta pelos elementos: *gerenciador de sessão*, *mecanismo de reflexão*, *construtor de serviço*, *servidor de serviço* e *implantador*. O *gerenciador de sessão* é responsável por controlar o estado de geração de serviços de cada componente solicitado pela biblioteca. O *mecanismo de reflexão* usa uma API de reflexão (Sobel e Friedman 1996) para extrair informações das classes dos componentes para construir os serviços. No caso do módulo Java, esse mecanismo utiliza a API de reflexão Java. O *construtor de serviço* e o *servidor de serviços* são os elementos mais importantes do módulo Java. O primeiro é responsável por transformar classes Java em serviços, enquanto que o segundo tem a tarefa de hospedar os serviços gerados para que possam ser utilizados. Estes dois elementos podem ser representados por Apache Axis2 (Apache 2009a), ferramenta amplamente utilizada em Java na construção de serviços. Axis2 oferece alguns recursos interessantes que facilitam o funcionamento do módulo Java. Por exemplo, Axis2 permite a configuração do *classpath* das aplicações Java transformadas em serviços. Este recurso é importante, pois permite que cada componente possua o seu próprio *classpath*, eliminando problemas de ambigüidade com componentes que possuem classes com nomes iguais. Outro recurso importante é a possibilidade de utilização de especificações oficiais de segurança de serviços web, como por exemplo, *WS-Security* (OASIS 2006). Esta especificação é importante para realizar o mecanismo de controle de acesso e registro de utilização. Por fim, o elemento *implantador* é responsável por preparar o componente para ser implantado no Axis2. Este elemento cria arquivos de configurações e utiliza a API de serviços disponível pelo Axis2 para realizar a comunicação.

O processo de geração de serviço foi definido de forma semi-automática a fim de garantir uma maior participação do produtor. Nesse processo, o produtor tem o poder de selecionar quais funcionalidades do componente devem ser transformadas em serviço. Na Figura 2.a é apresentado a tela de criação de serviços internos a partir de um componente na Brechó. Esta tela é apresentada ao usuário após o mesmo selecionar qual componente será transformado em serviço. No cadastro deve ser informada a linguagem de programação que o componente foi codificado para que a Brechó saiba com qual módulo se comunicar. Quando o usuário submete essas informações, a Brechó realiza a primeira comunicação com o módulo solicitando as operações do componente desejado. O módulo recebe a solicitação, armazena o componente temporariamente, para que seja novamente utilizado caso seja confirmado a geração do serviço, extrai as operações e

retorna para a Brechó, que são apresentadas em uma nova tela para selecionadas pelo usuário, de acordo com a Figura 2.b. Neste exemplo, um componente Java está sendo transformado em serviço, e as operações são mostradas em forma de árvore, onde os nós representam as classes e as folhas representam os métodos (as operações a serem selecionadas). No exemplo, os métodos seno e fatorial foram selecionados como as operações do serviço a ser criado. Após a seleção das operações, a Brechó se comunica novamente com o módulo encarregado para finalizar o processo de criação do serviço. No caso do módulo Java, com base no componente e nas funcionalidades selecionadas, o elemento *implantador* gera um arquivo de configuração do serviço que é passado, junto com os arquivos do componente, para o Axis2. No final, o documento de especificação do serviço (i.e., WSDL (W3C 2007)) gerado pelo Axis2 é enviado para a Brechó para finalizar a publicação do serviço.

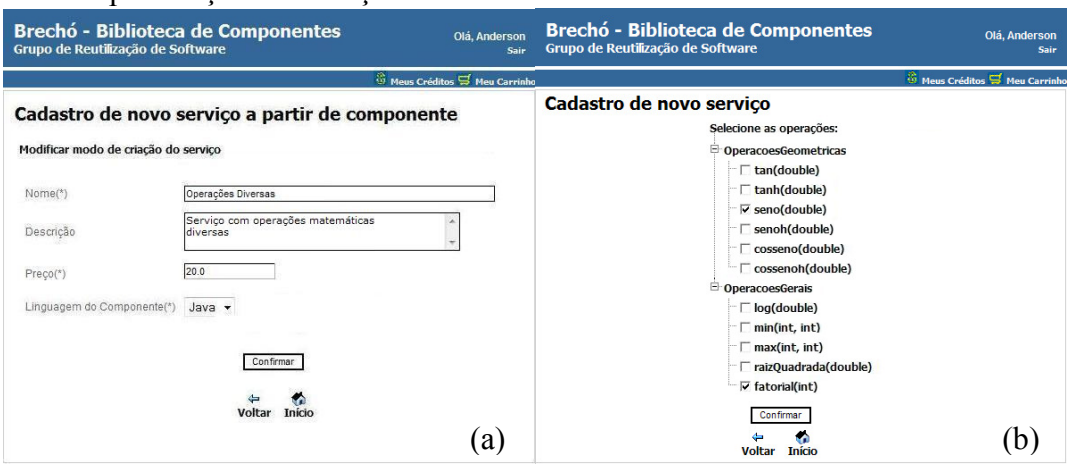

Figura 2 – Tela de cadastro de serviços internos simples na Brechó

2.2 Módulo Legado

O módulo Legado tem a responsabilidade de adaptar serviços externos, ou legados, para o contexto da Brechó. No entanto, realizar esta tarefa não é simples, uma vez que estes serviços estão hospedados em um servidor remoto, fora do alcance da Brechó. Uma solução para este problema é fazer com que essa adaptação seja feita de maneira indireta por meio da geração de um outro serviço que funcione como um *broker* (intermediador) entre os consumidores e o serviço original. A partir dessa intermediação, o serviço *broker* pode realizar as funcionalidades não contempladas pelo serviço externo, que é o caso dos mecanismos de controle de acesso e registro de utilização.

Cada serviço *broker* é construído com as mesmas características do serviço externo que ele representa, ou seja, um serviço *broker* provê os mesmos protocolos de comunicação e as mesmas assinaturas de operações. Essa similaridade é importante, pois permite que os serviços externos adaptados sejam usados da mesma forma que eles são usados fora da biblioteca. A principal diferença é a adição do protocolo de segurança WS-Security no serviço *broker* para auxiliar o mecanismo de controle de acesso.

A arquitetura do módulo Legado é composta pelos elementos: *servidor de serviço*, *adaptador de mensagem*, *gerador de serviço broker*. O *servidor de serviço* é responsável pela hospedagem dos serviços *brokers* que irão ser utilizados para representar os serviços externos adaptados. Da mesma forma que no módulo Java, a ferramenta Apache Axis2 foi utilizada para realizar essa função. O *adaptador de mensagem* tem a

tarefa de receber uma mensagem de requisição SOAP (W3C 2007), escrita no formato do serviço *broker,* e adaptá-la conforme a especificação do serviço externo. O *adaptador de mensagem* é configurado e plugado em cada serviço *broker* construído. Por fim, o *gerador de serviço broker* é responsável por construir os serviços *brokers* para cada serviço externo adaptado. Eles são construídos com base nos WSDL dos serviços externos, que contém as informações necessárias para criar um serviço semelhante, tais como definições de operações, protocolos e interfaces. Além disso, o *gerador de serviço broker* é responsável pela construção do *adaptador de mensagem* de acordo com o serviço externo a ser adaptado, bem como a construção dos mecanismos de controle de acesso e registro de utilização que serão utilizados pelo Axis2 para gerar o serviço *broker*.

Na Figura 3.a é ilustrado a tela de publicação de serviços externos na Brechó. O formulário de cadastro é semelhante ao do serviço interno, diferenciando-se apenas na substituição do campo *linguagem de programação* para o campo *WSDL*. Este campo solicita a URL do WSDL do serviço externo que o usuário deseja publicar. Quando o usuário submete o formulário, o serviço externo é publicado. No entanto, neste momento, a Brechó de maneira transparente para o usuário comunica com o módulo Legado para adaptar o serviço externo.

2.3 Módulo Orquestrador

Com os serviços internos e externos publicados na Brechó, um próximo passo é permitir que produtores publiquem novos serviços a partir da composição desses existentes na biblioteca. O módulo orquestrador foi concebido exatamente para realizar esta tarefa. Este módulo é responsável pela criação e hospedagem de serviços compostos a partir de documentos de especificação BPEL. Além disso, os serviços compostos gerados fornecem mecanismos de controle de acesso e registro de utilização, da mesma maneira que os outros serviços gerados ou adaptados pelos outros módulos.

A arquitetura do módulo Orquestrador é composta pelos elementos: *máquina orquestradora de serviços*, *servidor de serviço*, *implantador*, e *configurador*. A *máquina orquestradora de serviços* é responsável pela coordenação da execução de cada serviço que faz parte de um serviço composto, seguindo o que foi especificado no documento BPEL. O *servidor de serviço web* funciona em conjunto com a *máquina orquestradora de serviços* hospedando os serviços compostos. A ferramenta utilizada para representar estes dois elementos foi o Apache ODE (Apache 2009b). O *implantador* é utilizado pelo módulo para se comunicar com o *servidor de serviço* para realizar o processo de implantação de serviços. Este elemento é construído de acordo com o *servidor de serviço* e a *máquina orquestradora de serviços* utilizada. No caso do Apache ODE, o *implantador* foi construído de acordo com as suas características. Por último, o elemento *configurador* é responsável pela instalação dos mecanismos de controle de acesso e registro de utilização do serviço composto.

Na Figura 3.b é apresentado a tela de publicação de serviços internos compostos na Brechó. Este formulário é, também, similar aos outros formulários apresentados, diferenciando apenas pela inserção do campo *zip.* Este campo solicita os arquivos descritores do serviço composto, tais como documentos BPEL e WSDL. Ao submeter o formulário, a Brechó entra em contato com o módulo Orquestrador e solicita a geração de um novo serviço composto, passando os arquivos fornecidos pelo usuário. O módulo, por sua vez, aciona o elemento *configurador* para configurar o mecanismo de controle

de acesso e registro de utilização no serviço composto a ser criado. Esta configuração é feita por meio da geração de *scripts* que estão plugados ao serviço composto no momento de implantação no Apache ODE. Estes scripts serão acionados a cada requisição feita ao serviço. Depois de feita a configuração, o elemento *implantador* é ativado para realizar a implantação do serviço no Apache ODE. O *implantador* envia para o Apache ODE os descritores do serviço composto e os scripts gerados pelo elemento *configurador*. No final, o WSDL do novo serviço é enviado para a Brechó para ser publicado.

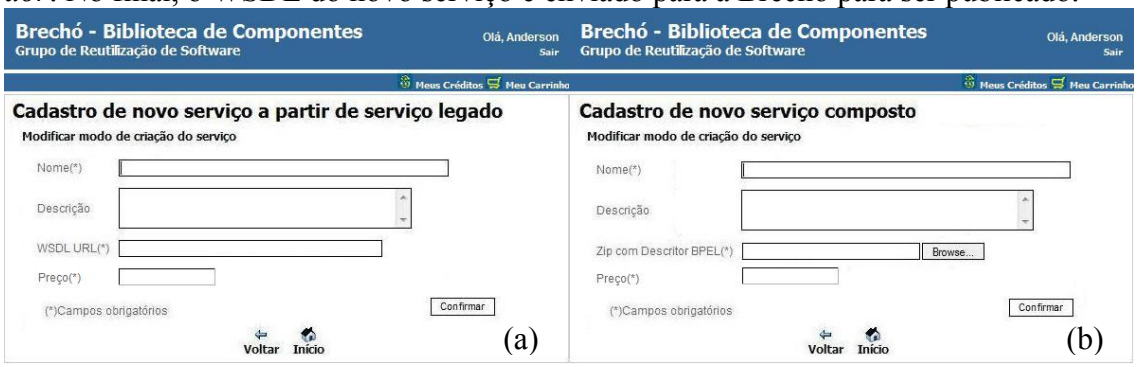

Figura 3-Tela de cadastro de serviços externos e serviços compostos na Brechó

3. Conclusões

Este artigo apresentou uma infra-estrutura para aplicação de serviços implementada na biblioteca de componentes Brechó. Esta infra-estrutura trata de questões sobre o controle de acesso e registro de utilização, além de apresentar mecanismos de geração de serviços, tanto a partir de componentes quanto a partir de descritores BPEL.

Existem alguns trabalhos na literatura que lidam com o problema apresentado. O trabalho de Lee *et al.* (Lee et al. 2004), por exemplo, apresenta um mecanismo de geração de serviço a partir de componentes em uma biblioteca. Este trabalho, porém, não fornece uma solução para o problema de componentes escritos em várias linguagens de programação, sendo limitado a apenas componentes codificados em Java ou C + +. Esta abordagem também não oferece uma solução para o problema de controle de acesso e não trata de questões relacionadas à geração automática de serviços compostos. Outro trabalho é o de Smith (Smith 2009), que utiliza reflexão para gerar serviços dinamicamente. No entanto, esta ferramenta não apresenta para o usuário uma previsão das possíveis operações que podem ser extraídas de uma determinada aplicação. Além disso, a ferramenta não gera automaticamente o WSDL dos serviços construídos. Por fim, existem algumas ferramentas comerciais de biblioteca de serviços (IBM 2009, Logic Library 2009), porém estas ferramentas não possuem mecanismos de geração automática de serviços, nem a partir de componentes ou por meio de descritores BPEL. Além disso, questões relacionadas á controle de acesso, também, não são tratadas.

Além do difundido estilo de serviços que utiliza a especificação SOAP e outras especificações correlatas, um outro estilo de serviços vem ganhando força, o estilo REST. Este estilo defende a construção de serviços a partir da utilização de operações básicas HTTP (GET, POST, DELETE, PUT), evitando a complexidade do uso de documentos XML, como é feito pelo protocolo SOAP. Devido à simplicidade do protocolo de comunicação, o estilo REST perde algumas funcionalidades importantes definidas para a especificação SOAP. Por exemplo, estes serviços não suportam a especifica-

ção de segurança WS-Security, que é fortemente utilizada na nossa infra-estrutura pelos mecanismos de controle de acesso e registro de utilização. Com isso, para viabilizar o funcionamento destes mecanismos, a nossa infra-estrutura não suporta serviços web do tipo REST. No entanto, pretendemos evoluir a infra-estrutura para suportar estes tipos de serviços também.

Uma outra limitação desse trabalho é que apenas o módulo especifico para a linguagem Java foi implementado. Entretanto, estamos analisando a possibilidade de implementar outros módulos como, por exemplo, .Net e C++. Além disso, o mecanismo de geração de serviços do módulo Java possui a restrição de ser compatível apenas com componentes que seguem o padrão JavaBean. A versão atual da biblioteca Brechó, incluindo esta infra-estrutura de serviços, está em funcionamento em http://reuse.cos.ufrj.br/brecho, onde está sob uma avaliação inicial acadêmica. Contudo, estamos planejando submeter a Brechó em uma avaliação mais sistemática em cenários reais.

Agradecimentos: Os autores agradecem à CAPES, FAPERJ e ao CNPq pelo apoio financeiro, e aos participantes do projeto Brechó.

Referências Bibliográficas

Apache, 2009a. Apache Axis2. Disponível em: http://ws.apache.org/axis2/.

Apache, 2009b. Apache ODE. Disponível em: http://ode.apache.org/.

IBM, 2009. WebSphere Service Registry and Repository. Disponível em: http://www-306.ibm.com/software/integration/wsrr.

Krueger, C. W., 1992, "Software reuse", *ACM Comput. Surv.*, v. 24, n. 2, p. 131-183.

Lee, R., Haeng-Kon Kim, Hae Sool Yang, 2004, "An architecture model for dynamically converting components into Web services". In: *11th Asia-Pacific Software Engineering Conference*, Busan, Korea., p. 648-654.

Logic Library, 2009. Logic Library Logidex. Disponível em: http://www.logiclibrary.com.

OASIS, 2006, *Web Services Security (WS-Security) 1.1*.

OASIS, 2007, *Web Services Business Process Execution Language (WSBPEL) 2.0*.

Papazoglou, M., 2003, "Service-oriented computing: concepts, characteristics and directions". In: *Fourth International Conference on Web Information Systems Engineering*, Rome, Italy, p. 3-12.

Sametinger, J., 2001, *Software Engineering with Reusable Components*. 1 ed. Springer.

Smith, M., Engel, M., Friese, T., Freisleben, B., 2006, "Security issues in on-demand grid and cluster computing". In: *Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops*, Singapore, p. 14-24.

Smith, Z., 2009. Applied Reflection: Creating a dynamic Web service to simplify code. Disponível em: http://downloads.techrepublic.com.com/download.aspx?docid=264551.

Sobel, J. M., Friedman, D. P., 1996, "An introduction to reflection-oriented programming". In: *Proceedings of Reflection'96*, San Francisco, CA, USA, p. 263-288.

Szyperski, C., 1997, *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Professional.

W3C, 2007, *Web Service Description Language (WSDL) 1.1*.

W3C , 2007, *Simple Object Access Protocol (SOAP) 1.2 (Second Edition)*.

Werner et al., 2007, "Brechó: Catálogo de Componentes e Serviços de Software". In: *SBES, Sessão de Ferramentas*, João Pessoa, Brasil, p. 24-30.