Motor de inferências em plataforma OSGi

Ding Yih An, Juan X. E. A. Calles, Karin Komati, Maxwell E. Monteiro

Coordenadoria de Informática – Instituto Federal do Espírito Santo (IFES) Campus Serra, Rodovia ES–010, Km 6.5 – 29.164-231 – Serra – ES – Brasil

{callesjuan,sidnha}@gmail.com; {kkomati,maxmonte}@ifes.edu.br

Abstract. *Most of the Ontology-based applications developed nowadays use OWL files for representing their ontology models, also they embed reasoners so that they can validate these OWL files. This work intends to show the benefits of providing reasoning as a service, also, briefly presents the steps for an implementation.*

Resumo. *A maioria das aplicações de base ontológica desenvolvida nos dias de hoje, usam arquivos OWL para representar seus modelos de ontologias. Elas também incorporam raciocinadores para que possam validar esses arquivos OWL. Este trabalho pretende mostrar os benefícios da prestação de raciocínio como um serviço, também apresenta brevemente os passos para a sua implementação.*

1. Introdução

Em definição curta, uma ontologia é uma especificação de uma conceituação (Gruber, 1993) que expressa os conceitos e as relações de um determinado domínio de forma não ambígua. Geralmente são representadas computacionalmente em formato OWL (*Web Ontology Language*), o que permite a definição de regras baseadas em lógica descritiva, chamadas de axiomas (Lindörfer, 2010). É possível fazer validações semânticas em arquivos OWL através de ferramentas chamadas de raciocinadores (*reasoners*), tais como: verificar a consistência da ontologia e inferir novas informações a partir da mesma. Os *reasoners* são utilizados em momentos específicos do ciclo de vida da aplicação: início de operação e nas atualizações das ontologias utilizadas.

Geralmente, o ferramental para a construção de aplicações baseadas em ontologias não suporta, diretamente, o paradigma de computação como serviço (Hewitt, 2009). Nesse contexto de computação como serviço, vale destacar o OSGi (*Open Services Gateway Initiative*), que é um conjunto de especificações que definem um sistema de componentes dinâmicos para Java, levando em consideração questões de empacotamento, versionamento, implantação, publicação e descoberta de serviços. O conceito fundamental que permite o funcionamento de um sistema desse tipo é a modularidade e o uso de interfaces de acesso a serviços. Módulos são componentes menores e reutilizáveis, aqui denominados de *bundles*, que escondem as suas implementações de outros componentes durante a comunicação por meio de serviços. Esses *bundles* são capazes de publicar serviços dinamicamente, além de descobrir e requisitar serviços disponibilizados por outros *bundles*. Uma característica importante desta especificação é que qualquer *bundle* pode ser executado em qualquer implementação do OSGi. (OSGi Alliance, 2012).

De forma geral, sistemas baseados em ontologias tem o *reasoner* embutido em si, o que aumenta a sua complexidade e a quantidade de recursos necessários para sua

execução. Em uma plataforma OSGi, por exemplo, estas questões se tornam evidentes quando se imagina que cada *bundle* envolvido em uma aplicação de ontologia deveria incluir em si uma instância distinta de um determinado *reasoner*.

O presente trabalho propõe a criação de um motor de inferências que visa atender o crescimento de sistemas distribuídos que fazem uso de ontologias, minimizando o trabalho de se ter que carregar múltiplas instâncias do *reasoner* dentro de seus módulos, já que o uso desses *reasoners* é limitado a certos momentos da sua operação. Desta forma, promove-se o reuso e diminui-se a complexidade da gestão de instâncias e artefatos relacionados, aproveitando as facilidades que uma plataforma OSGi oferece (Monteiro, 2010).

2. Solução Proposta

O *reasoner* escolhido para implementação deste trabalho foi o Pellet (Clark & Parsia, 2011) por ser um software consolidado e o primeiro a suportar inferências em artefatos OWL. A implementação OSGi escolhida foi o Eclipse Equinox (The Eclipse Foundation, 2013), por ser a implementação de referência da OSGi Alliance, a partir da revisão 4 da especificação OSGi. A utilização de outro *reasoner,* ou implementação OSGi, não deve afetar o resultado da solução proposta.

Para demonstrar o fornecimento de serviços semânticos na verificação de consistência e inferência de novas informações, foi criado um *bundle* OSGi que disponibiliza as funcionalidades do motor de inferência Pellet. Os serviços de raciocínio relevantes identificados na implementação são os de:

- consistência - certifica que a ontologia não contém quaisquer fatos contraditórios;
- classificação - calcula as relações de cada subclasse com as outras classes para descobrir todas as heranças existentes e criar uma hierarquia completa a partir disso e;
- realização - encontra a classe específica à qual cada um dos indivíduos pertence.

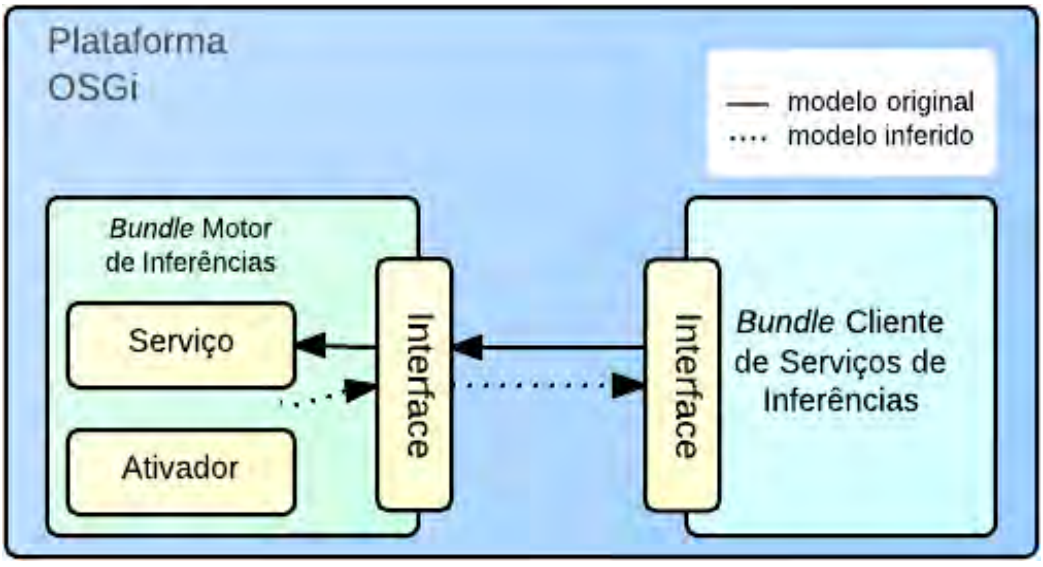

Figura 1 - Cliente acessando os serviços do *bundle* Motor de Inferências.

O modelo da solução é constituída por três partes, conforme visto na Figura 1:

- interface - que descreve os serviços oferecidos com os seus respectivos parâmetros;
- serviço - implementa os serviços de raciocínio descritos previamente na interface e;
- ativador - responsável por registrar e disponibilizar o serviço implementado, ou seja, pelo ciclo de vida do *bundle*.

O funcionamento se dá pela criação do *bundle* descrito acima, composto por interface, serviço e ativador, e em seguida especificam-se no arquivo de manifesto as interfaces dos próprios serviços que ele exportará. Finalmente, ele é registrado na plataforma através do ativador. As informações contidas no arquivo de manifesto serão disponibilizadas para os demais *bundles*. Para o consumo dos serviços, é preciso que clientes importem em seus arquivos de manifesto as interfaces do motor de inferências Pellet e referenciem os serviços no seu próprio ativador.

A partir da arquitetura definida, o cliente é capaz de utilizar os serviços citados anteriormente, fornecendo um modelo ontológico (modelo original) como entrada (como mostrado pelas setas contínuas na Figura 1) e tendo como retorno o modelo inferido (conforme mostrado pelas linhas tracejadas na Figura 1). A passagem do modelo como parâmetro de entrada é representada pelo seu valor literal, devido à sua simplicidade, garantindo a transparência de localização e minimizando o trabalho de conversão para um protocolo de aplicação.

Serviços de inferências são custosos, para solucionar esta questão, propõe-se a adoção de um *pool* de conexões no motor de inferências, a fim de limitar o seu acesso e consequentemente preservar a disponibilidade deste recurso. Através da definição da quantidade máxima de requisições, será possível que os serviços de inferências operem dentro dos limites que o *hardware* da plataforma permite. Só haverá aumento no número de acessos depois que novos recursos de *hardware* forem adicionados e a quantidade máxima de requisições ao *pool* for aumentada, provendo um estágio inicial em relação à escalabilidade.

3. Experimentos e Resultados

A presente definição do motor de inferências como serviço está sendo aplicada em um sistema de gerência autônoma de redes, e encontra-se implementada em linguagem Java, utilizando as ferramentas descritas acima.

Durante os testes, um modelo didático de ontologia de redes foi desenvolvido na ferramenta Protégé (Stanford Center for Biomedical Informatics Research, 2013). Um *bundle* cliente foi criado para consumir os serviços propostos. A partir do cliente foram realizadas as chamadas aos serviços desenvolvidos sobre a ontologia de teste e os resultados das inferências foram salvos. Para uma comparação, as mesmas inferências foram realizadas na ferramenta Protégé. Finalmente, os resultados do *bundle* cliente e os da ferramenta Protégé foram comparados e demonstraram ser equivalentes, comprovando a consistência das informações encontradas.

A solução trouxe como resultado a diminuição da complexidade em lidar com múltiplas instâncias de *reasoners* e seus artefatos, uma vez que foi abstraído o acesso à

ferramenta de inferências Pellet, permitindo a gestão mais simplificada de seus recursos. Além disso, promoveu-se o reuso através da definição de um componente responsável por serviços de inferência, ao invés da replicação desses serviços em cada aplicação.

4. Considerações Finais

O trabalho apresenta os benefícios da modularidade de motores de inferência em plataforma OSGi para diminuição da complexidade e reuso em sistemas que requerem a manipulação de artefatos semânticos. É importante destacar que a centralização de serviços de inferências acrescenta um ponto de falha para sistemas baseados em ontologias, uma vez que a indisponibilidade do motor de inferências compromete o funcionamento de todos os requisitantes do serviço. Por isso, a arquitetura apresentada deve ser aprimorada visando um melhor desempenho e qualidade dos resultados.

Um trabalho correlacionado é o OWLTools, que efetua inferências do tipo checagem de consistência, descoberta de subclasses ou superclasses, dentre outras. O que difere desta abordagem é o fato de tratar-se de uma API independente destinada a oferecer suporte completo para artefatos OWL, com funcionalidades além dos *reasoners,* enquanto a solução proposta destina-se a oferecer apenas serviços de inferências em um ambiente modular.

Futuros trabalhos podem contemplar a possibilidade de suportar a adoção de outros *reasoners* além do Pellet sob escolha parametrizável. Também há possibilidade de exportar as funcionalidades de um motor de inferências para outras plataformas além de OSGi, através da definição de outros artefatos modulares específicos que implementem a mesma interface de serviço. Por exemplo, utilizar o acesso via WebServices ou RMI (*Remote Method Invocation*). Por fim, é possível aplicar a abordagem proposta neste trabalho para a implementação de outros módulos de igual comportamento, como, por exemplo, um motor de persistência de ontologias.

Referências

Clark & Parsia (2011). “Pellet Help”, http://clarkparsia.com/pellet/docs/, Dezembro.

The Eclipse Foundation (2013). “Equinox Resources”, http://www.eclipse.org/equinox/resources.php, Janeiro.

Gruber, T. (1993). “A Translation Approach to Portable Ontology Specifications”. Knowledge Specifications, v. 5, n. 1, p. 199-220, Junho.

Hewitt, E. (2009). Java SOA Cookbook, 1ª edição.

Lindörfer, F. (2010). “Semantic Web Frameworks”, http://cswwwarchiv.cs.unibas.ch/lehre/hs10/cs341/_Downloads/Workshop/Reports/2010-HSDIS-F_Lindoerfer-Semantic_Web_Frameworks-Report.pdf., Dezembro.

Monteiro, M. E. (2010). “Uma Proposta de Serviços Semânticos Relacionada ao Autogerenciamento em Redes Ópticas de Transporte”. Tese (Doutorado em Engenharia Elétrica – UFES).

OSGi Alliance (2012). “The OSGi Architecture”, http://www.osgi.org/Technology/WhatIsOSGi, Dezembro.

Stanford Center for Biomedical Informatics Research (2013). “Protégé user documentation”, http://protege.stanford.edu/doc/users.html, Dezembro.