ITA Semantic Library - Uma Biblioteca para a Construção de Aplicações Semânticas em uma Arquitetura de C2

Alexandre de Barros Barreto, Edgar Toshiro Yano, José Parente de Oliveira
Instituto Tecnológico de Aeronáutica
Praça Marechal Eduardo Gomes 50, Vila das Acácias
12228-900 - São José dos Campos - SP
e-mail: kabart@gmail.com / yano@ita.br / parente@ita.br

***Resumo*—As novas arquiteturas que visam atender às aplicações de Comando e Controle devem ter como característica a capacidade de manipular informações dinâmicas e altamente distribuídas, usando para isso um paradigma orientado a serviço. Para aumentar a robustez e escalabilidade desse tipo de arquitetura, o uso de semântica deve ser considerado. Nos trabalhos encontrados na literatura, o uso de semântica em serviços está normalmente relacionado a descrição dos mesmos, resolvendo a problemática da sua descoberta. Porém, em uma abordagem mais completa, seria necessário que essa tecnologia estivesse dentro da construção do próprio serviço. O problema dessa abordagem é a pouca documentação existente que apresente como essa atividade deve ser realizada, bem como a complexidade do uso das bibliotecas existentes. O presente trabalho apresenta uma biblioteca desenvolvida para simplificar o processo de desenvolvimento de um serviço semântico. Para validar a biblioteca desenvolvida, duas métricas quantitativas foram utilizadas: a quantidade de classes necessárias para representar um conceito e o número de código-fonte necessário para implementar um serviço. O experimento que permitiu concluir a simplicidade de uso da ferramenta foi desenvolvido através da construção de um serviço semântico usando o framework Protégé e a ITA Semantic Library.**

***Palavras-chaves*—Web Semântica, Comando e Controle, Biblioteca.**

I. Introdução

A busca pela superioridade da informação em conflitos modernos é uma realidade [1] [2]. Essa busca é baseada na construção de sistemas de comando e controle (C2) robustos capazes de permitir o exercício da autoridade e da direção por um comandante, visando que o mesmo cumpra a missão designada [3].

Para estruturar esses sistemas, Forças Armadas de diferentes países, adotaram a arquitetura orientada a serviços (SOA) como base de seus sistemas [1] [4], o que traz uma infinidade de possibilidades, em especial quando analisada a integração e cooperação de informações entre os diversos componentes de um teatro de operações.

Para potencializar a funcionalidade desses sistemas, é desejável o uso de informações semânticas para descrever e implementar os serviços, uma vez que aumenta a capacidade de recuperação das informações existente na infraestrutura de C2 implantada [5], bem como a descoberta de informações e comportamentos não mapeados durante a concepção do sistema [6]. Nesse cenário, o uso de semântica pode ser realizado de duas formas: na descrição e na construção do serviço.

Porém o uso de semântica em aplicações é algo ainda bem restrito, em virtude da existência de componentes, frameworks e bibliotecas, que, em sua maioria, são muito complexas e pouco documentadas.

O presente trabalho tem por finalidade apresentar como a biblioteca ITA Semantic Library pode simplificar o processo de construção de aplicações semânticas. Como restrição de escopo, não será explorado como a mesma pode ser usada para descrever um serviço semântico, uma vez que existe documentação adequada sobre esse processo [7], mas sim o procedimento de inserção da semântica no corpo de uma aplicação.

Para alcançar esse objetivo, este artigo está organizado da seguinte forma: a seção 2 apresenta o que é um Serviço Web Semântico. Na sequencia será abordado como usar semântica em uma aplicação de Comando e Controle. Na seção 4 serão apresentadas as principais funcionalidades da biblioteca desenvolvida. Na seção 5, um estudo de caso foi desenvolvido, demonstrando o uso da biblioteca e validando o mesmo através de métricas definidas. Para finalizar o trabalho algumas considerações finais e aspectos acerca de limitações e oportunidades de continuidade das pesquisas desenvolvidas foram delineados.

II. Web Services e Web Services Semânticos

Uma arquitetura orientada a serviço (*Service-Oriented Architecture - SOA*) é um estilo arquitetural de software cujo princípio fundamental é a implantação das funcionalidades de um sistema através de interfaces bem definidas, o que é chamado de serviços [8] [9]. Uma

aplicação que adote SOA como paradigma deverá possuir as propriedades de baixo acoplamento e interfaces bem definidas.

Em uma arquitetura SOA, o principal conceito existente é o serviço. Um serviço pode ser definido como uma função independente, sem estado, que aceita uma ou mais requisições e devolve uma resposta através de uma interface padronizada e bem definida [8] [9].

O funcionamento dessa arquitetura necessita que o serviço, para ser usado, seja divulgado, possibilitando aos diversos elementos da rede identificá-lo. Dessa forma, existem três componentes básicos nessa arquitetura: um consumidor, um prestador do serviço e um repositório onde os mesmos são publicados e encontrados.

Uma implementação de uma arquitetura SOA é o *Web Service - WS*, que consiste em um sistema de software capaz de suportar interoperabilidade entre máquinas sobre uma rede, usando protocolos baseados em padrões da Internet, geralmente HTTP [10] com serialização XML [11], para descrever a interface, a busca e o uso do serviço [12].

Para descrever um serviço, um provedor usa uma linguagem baseada em XML chamada *Web Services Description Language - WSDL* [13]. Um arquivo WSDL tem por finalidade apresentar de forma clara as interfaces do serviço, ou seja, como ele é acessado (binding), em que endereço ele se encontra, que classe ele é, bem quais são os parâmetros de entrada e retorno.

Para que o serviço possa ser acessado, um WS pode usar o protocolo *SOAP - Simple Object Access Protocol* [15] para transportar as requisições e respostas a ele. Uma mensagem SOAP é um envelope XML que usa uma mensagem HTTP ou RPC como protocolo de transporte e pode transportar qualquer tipo de conteúdo MIME em seu corpo. Uma outra alternativa para possibilitar esse transporte é o uso da arquitetura *REST - Representational State Transfer*, que tem por vantagem em relação a SOAP o fato de ser mais simples de usar e ter seus resultados mais simples de serem apresentados a um usuário humano. Porém, o protocolo SOAP é mais simples de ser consumido pelos serviços [14].

Já que uma arquitetura orientada a serviço é distribuída, nem todos os clientes saberão onde os mesmos estão localizados, nem como os invocar. Para endereçar essa questão, um repositório deve ser disponibilizado. Uma alternativa para a implantação desse repositório é através do protocolo *Universal Description Discovery and Integration - UDDI* [16], cuja finalidade é definir como os registros devem ser feitos e como as buscas nesse diretório devem ser realizadas.

Os serviços podem ser descritos de várias formas diferentes, o que torna a busca em um diretório UDDI um processo complexo e muitas vezes improdutivo, já que muitas das consultas podem não retornar nenhum resultado útil para o consumidor. Para resolver esse problema, foi definida uma estrutura de dados XML, na qual todos os objetos devem ser descritos, o *t-model* [12]. O t-model cria uma taxonomia para os serviços, classificando-os em tipos, possibilitando que as buscas possam ser feitas com base em uma categoria, como por exemplo namespace, locais geográficos, tipos de negócios, entre outros.

Devido a característica de baixo acoplamento, para realizar uma tarefa pode ser necessária a concatenação de diversos serviços, visando a criação de um serviço maior, o que é chamado de composição. Um exemplo de composição seria um sistema de venda de passagens que utiliza dois outros serviços para realizar sua tarefa, um que verifica a disponibilidade de assentos em um voo específico e outro que analisa se o usuário possui crédito em seu cartão de crédito para realizar a compra. Apesar desses dois serviços serem necessários, para o usuário ele interage com um sistema único, onde ele passa o cartão e os dados da viagem e o sistema retorna a passagem desejada.

Conforme apresentado em [12], a arquitetura SOA apresenta alguns problemas. O primeiro consiste na descoberta dos serviços, uma vez que o cliente tem que saber exatamente como invocar o mesmo, já que o processo de registro e busca em diretório são realizados de forma sintática. Por exemplo, se a busca for por um serviço que retorne um conjunto de funcionários, mesmo que o diretório possua um serviço que devolva um conjunto de empregados, ele não será capaz de retornar nenhuma informação para o consumidor do serviço.

Outro problema diz respeito a possibilidade de composição automática de serviços. Quando um serviço do tipo (A) não estiver disponível, mesmo exista um outro bem parecido, ele não será capaz de encontrá-lo, já que o diretório usa parâmetros fixos e descritos sintaticamente para mapear qual serviço deve ser usado.

Para contornar esses problemas, uma estratégia é usar informações semânticas para descrever os serviços de um WS, o que é chamado de *Semantic Web Services - WSS*, usando para isso uma ontologia de domínio específica para essa tarefa.

O conceito ontologia pode ser entendido como um conjunto de conceitos e seus relacionamentos, que tem por finalidade descrever e possibilitar o entendimento de um domínio de aplicação. Segundo [17], uma ontologia pode ser entendida como uma especificação formal e explícita de um conceito.

Apesar de existirem vários padrões para implementar serviços semânticos, esse trabalho focará apenas no *OWL-S* [18], por este poder ser aplicado em múltiplos domínios de negócios, não sendo restritivo como algumas tecnologias correlatas [19]. Esse protocolo cria um

conjunto básico de classes e propriedades para descrever serviços.

Para que um serviço seja descrito semanticamente é necessário informar o que ele faz, como faz e como invocar o mesmo. Para realizar essa tarefa, o OWL-S cria uma ontologia independente de domínio (*service.owl*) das quais todos os serviços devem ser originados [12].

Para definir o que o serviço faz é usado o *Service-Profile (profile.owl)*, que define quais são as informações gerais e uma classificação do serviço, o seu responsável, uma descrição funcional, que contém as entradas e saídas, as pré-condições e os efeitos esperados, bem como alguns parâmetros não funcionais como segurança, qualidade, entre outros. É o perfil que é usado pelas ferramentas de buscas para localizar um serviço semântico.

Uma vez selecionado o serviço, através das informações existentes em seu perfil, para saber como o mesmo é implementado, são utilizadas as informações constantes no *ServiceModel*, que o detalha mais, permitindo ao cliente ter uma compreensão de como o mesmo funciona, consequentemente, o que ele faz internamente, capacitando ao mesmo concluir se a funcionalidade desejada é realmente implementada pelo WSS analisado. Essa capacidade permite que sejam implementados sistemas de composição dinâmicas, bem como o monitoramento do serviço.

Para finalizar, após o cliente descobrir se o serviço selecionado é o desejado, ele necessita saber como invocar o mesmo. Para realizar essa tarefa é usado o *ServiceGrounding*, que mapeia a ontologia para as informações concretas de acesso contidas no WSDL.

A descrição de um WS de forma semântica possibilita que o mesmo possa ser encontrado de forma mais simples, bem como viabilizar aos mecanismos de busca uma forma de retornar informações para os clientes com base em seus parâmetros comportamentais (qualidade, custo, segurança, entre outros). Apesar disso, uma abordagem mais completa seria o uso de uma ontologia de domínio na implementação dos serviços concretos disponibilizados pelo WS. Essa abordagem permitiria o uso de mecanismos de dedução (inferência) e raciocínio (reasoning) para classificar os indivíduos e as propriedades dos serviços, dando a aplicação um poder de adaptação baseada em contexto em tempo de execução, o que é bastante complexo de fazer em uma aplicação implementada de forma sintática.

III. O Uso de Semântica em Aplicações de C2

Para ser possível uma análise sobre o impacto de um sistema semântico em um ambiente de C2, se faz necessário um detalhamento acerca de seu ambiente operacional.

Em [20] é apresentada uma iniciativa do governo americano em definir o futuro sistema de combate utilizado pelas Forças Terrestre dos Estados Unidos, o chamado *Future Combat System - FCS*. O programa FCS tem por objetivo interligar soldados, sistemas de armas e de C2 por meio de uma arquitetura de rede móvel *ad-hoc* que permitirá níveis de interoperabilidade, consciência situacional compartilhada e a capacidade de executar operações de missão altamente sincronizadas.

Em [21] é apresentado o ambiente de C2 em camadas, onde cada uma delas possui características diferentes, fazendo com que a análise sobre os sistemas sejam realizadas de forma segmentada. Apesar de o supracitado trabalho apresentar uma diminuição de serviços nos níveis táticos, projetos como o FCS demonstram que a arquitetura SOA será necessária em todos os níveis para o funcionamento das aplicações de C2.

Em [4] é apresentada uma proposta de framework que tem por finalidade possibilitar que um componente aéreo (Força Aérea) possa interoperar com agências diversas quando em um cenário que necessite uma operação conjunta. Esse framework usa uma abordagem orientada a serviço. A abordagem adotada no supracitado trabalho é semelhante a de [4], onde existe uma segmentação baseada em níveis, onde é analisado apenas o estratégico. Esse trabalho, segundo o autor, é justificado por possuir alinhamento com uma necessidade do Ministério da Defesa (MD) do Brasil, que buscas soluções orientadas a serviço e de soluções baseadas em semântica.

Uma outra abordagem para o uso de semântica em aplicações de defesa é na seleção de componentes em um teatro de operações de forma dinâmica. Essa abordagem é usada em [28], onde é realizada uma modelagem de ambientes de simulação que necessitam de composição dinâmica de serviços, como na reconfiguração automática de federações com base em eventos.

Em todas as abordagens anteriormente apresentadas, a arquitetura orientada a serviço é referenciada como o meio mais adequado para atender as necessidades dos diversos nós componentes de um teatro de operações, pois permite que as diversas entidades envolvidas, nos seus mais variados níveis de atuação, possam, de uma forma bastante transparente, localizar e acessar os serviços necessários para a realização de sua tarefa. Porém, conforme comentado anteriormente, para otimizar a acuracidade desses algoritmos, o uso de semântica deve ser considerado.

Para realizar a implantação desse cenário com semântica, uma ontologia única de domínio deve existir, caso contrário seria necessário a presença na rede de um agente capaz de realizar alinhamento de ontologias diferentes, gerando um mapeamento entre os conceitos existentes em cada uma delas, possibilitando a integração das mesmas [22].

Conforme comentado na *Seção II - Web Services e Web Services Semânticos*, o uso de semântica em

um WS pode ser feito de duas formas. Na primeira abordagem, usa-se semântica apenas para descrever o serviço, enquanto na segunda, e mais completa, essa tecnologia é usada não só para descrever o WS, mas também para implementar os serviços.

Para possibilitar que uma ontologia seja acessada por uma aplicação, duas estratégias podem ser usadas. A primeira consiste em converter o modelo de domínio (representado por uma linguagem como OWL) em um modelo de classes, o que faria com que os mecanismos de inferência e reasoning existentes tivessem que ser implementados no modelo de classes, tornando-o pouco generalista e extensível.

Uma segunda abordagem seria usar um componente semântico que possibilitasse a integração do modelo de classes ao modelo ontológico existente. Essa abordagem, apesar ser mais complexa, constrói uma aplicação mais extensível e com capacidade de utilizar todos os recursos suportados pela linguagem de descrição da ontologia.

Conforme apresentado em [20] [21] [22], um sistema de defesa é constituído por elementos heterogêneos e trabalha com informações que mudam a todo o tempo, em virtude do dinamismo ambiente no qual ele está inserido.

Nesse cenário, a semântica pode contribuir de forma bastante decisiva, através do uso de regras e de mecanismos de classificação baseado em inferências. Com o uso de regras é possível que uma informação possa ser classificada em tempo de execução, sem a necessidade de recompilação do código fonte, bastando que um atributo do conceito utilizado mude ou o contexto no qual ele está inserido. Por exemplo, considere um sistema de gestão de tráfego aéreo que necessite dividir as aeronaves em setores usando para isso um conjunto de fatores dinâmicos [23]. Caso, um novo fator não mapeado durante a concepção do sistema ocorra, o sistema não poderia tratar essa informação sem uma manutenção em seu código-fonte. Com o uso de regras esse problema é possível de ser tratado em tempo de execução, bastando para isso a modificação na ontologia desse parâmetro, fazendo com que toda a aplicação seja ajustada.

O uso de semântica também permite que buscas de informações possam ser feitos sem que os critérios de buscas e os conceitos procurados tenham sido definidos durante a concepção do sistema. Um exemplo de uso de buscas semânticas pode ser visto em um sistema que, ao ser modelado, insere um conceito militar generalista como base da informação. Porém, por uma circustância especial, é necessário recuperar os militares da Força Aérea e não todos os militares. Nesse cenário, uma linguagem como a SPARQL [24] pode endereçar essa questão.

IV. ITA Semantic Library

Para prover uma capacidade semântica aos sistemas apresentados em [4] e [20] é necessário que os serviços além de terem capacidade de serem descritos semanticamente, também sejam construídos utilizando tal paradigma. Uma vez que a descrição semântica é uma tarefa que faz parte do processo de modelagem e existe documentação a respeito [7], o presente trabalho focou em criar um ambiente que permitisse a construção de serviços semânticos, possibilitando a manipulação de suas informações e a realização de operações sobre as mesmas, como consultas e inferências.

Para essa tarefa, a primeira estratégia adotada foi a utilização do *Jena* [25]. O *Jena* é um framework Java para construção de aplicações semânticas na Web. Ele oferece um ambiente de programação para ontologias descritas nas linguagens RDF, RDFS e OWL, além de possuir suporte a consultas baseadas em linguagem SPARQL e incluir um motor de inferência baseado em regras.

Apesar do Jena ser um framework bastante robusto e possuir um recurso de persistência bastante simples de ser utilizado, a manipulação de indivíduos é bastante complexa, em especial quando se deseja trabalhar com Object Properties, fazendo com que seu uso tenha que ser realizado por especialistas em semântica, fazendo com que fosse buscada uma abordagem mais simples.

A segunda abordagem adotada foi usar a *API (Application Programming Interface)* da própria ferramenta utilizada para a modelagem e construção das informações semânticas do WS, o *Protégé* [26]. O Protégé é um editor e um framework de construção de ontologias baseado em código aberto. A supracitada ferramenta é baseada em Java, é extensível, e fornece um ambiente modular que a torna uma base flexível para prototipagem rápida e desenvolvimento de aplicações semânticas.

O Protégé possui uma função de exportação dos conceitos modelados através de seu editor para Java. Essa função cria um conjunto de classes e interfaces que representam todos os conceitos envolvidos na ontologia trabalhada (classes, *datatype properties*, *object properties*, regras, indivíduos, etc). Além disso, ela gera uma classe especial chamada *MyFactory*, cujo objetivo é possibilitar a construção dos diversos indivíduos que existem na ontologia. Para ser possível atribuir valores as propriedades existentes nos indivíduos recém-criados, é necessário o uso dos métodos *add()* ou *set()* existentes nos indivíduos da ontologia.

O uso do Protégé, apesar de ser mais simples do que a abordagem anterior, o Jena, ainda apresenta bastante complexidade quando se precisa unir os diversos conceitos e mecanismos existentes na tecnologia semântica para a construção de um sistema, em especial em virtude de a documentação existente ser bastante segmentada

e não apresentar uma visão unificada do processo de construção de um sistema semântico.

De posse dessas limitações, foi desenvolvida uma biblioteca que tinha por finalidade fornecer a um desenvolvedor uma forma prática e simples de manipular informações semânticas, viabilizando seu uso em sistemas reais, o que foi chamado de *ITA Semantic library*, que cria um facede [27] para acesso as funcionalidades existentes no Protégé, bem como dos demais componentes que ele faz uso, simplificando dessa forma o acesso ao desenvolvedor das funcionalidades semânticas contidas no supracitado framework.

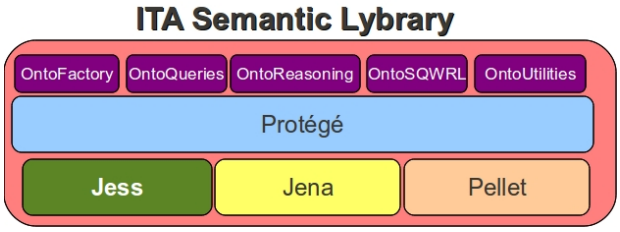

Figura 1. ITA Semantic Library.

Para simplificar o processo de desenvolvimento de sistemas semânticos, a ITA Semantic library faz uso de cinco classes principais: *OntoUtilities*, *OntoFactory*, *OntoReasoning*, *OntoQueries* e *OntoSQWRL*.

Na classe de utilidades da biblioteca, a *OntoUtilities*, é possível através de um arquivo OWL contendo a ontologia desejada, carregar a mesma em memória, bem como gerar a representação dessa ontologia em um modelo de classes Java. Além disso, é essa classe que possui o método de persistência da ontologia carregada em memória em um arquivo no formato XML.

Para ser possível a manipulação de indivíduos (adicionar, remover e recuperar) pela aplicação, deve-se usar a classe OntoFactory.

A classe OntoQueries permite ao desenvolvedor, através da linguagem SPARQL, a construção de consultas semânticas na ontologia em tempo de execução.

Outra classe importante é a OntoReasoning que permite o uso de um mecanismo de reasoning no sistema desenvolvido, viabilizando que indivíduos contidos na ontologia sejam classificados e inferidos em tempo de execução.

Para finalizar, uma das classes mais poderosas da biblioteca é a *OntoSQWRL*. É através dela que um mecanismo de regras pode ser usado. Para construir regras e aplicá-las na ontologia é usado a linguagem *SQWRL - Semantic Query-Enhanced Web Rule Language* [28], que tem por finalidade definir um meio de aplicar regras construídas usando *SWRL - Semantic Web Rule Language* [29].

V. Construção de um Sistema de C2 Usando a ITA Semantic Library

Conforme apresentado anteriormente, a ITA Semantic Library foi desenvolvida para simplificar o processo de construção de sistemas semânticos, possibilitando que as vantagens do uso dessa tecnologia fosse facilmente incorporado em aplicações de defesa.

Para validar o aumento de simplicidade na codificação dos sistemas foram usadas duas métricas quantitativas. A primeira diz respeito ao número de classes necessárias para a construção de serviços semânticos. O segundo foi o número de linhas de códigos necessárias para a implantação desses mesmos serviços. Apesar de ser desejável o uso de métricas qualitativas [30], em virtude da restrição de avaliar o uso da biblioteca em cenários reais, em virtude da biblioteca ter sido desenvolvida em um ambiente acadêmico, essa abordagem não foi utilizada.

Para realizar a aplicação das métricas, um estudo de caso foi construído com base em um dos problemas atuais de Comando e Controle, o uso de serviços semânticos como meio de permitir que diversos agentes interajam em uma situação de calamidade (Forças Armadas, Forças de Segurança, Organizações não governamentais - ONG, etc), mesmo possuindo arquiteturas de informação diferentes [4]. Esse tipo de cenário é bastante comum atualmente nas Forças Armadas brasileiras [31] [32] [33].

Uma vez que a proposta era apenas demonstrar o uso da biblioteca e não modelar um sistema para ser usado em um cenário real, diversas simplificações foram usadas e o resultado final da modelagem pode ser visto na figura 2.

Figura 2. Ontologia do Cenário de Ajuda Humanitária.

De posse de um entendimento dos conceitos envolvidos no cenário estudado, torna possível apresentar como usar a biblioteca. O primeiro método que deve ser usado é o método estático *OntoUtilities.initOntologySystem(String OntologyURI)*.

Esse método, ao ser chamado realiza duas tarefas. A primeira consiste em verificar se existem as classes e interfaces Java que representam a ontologia passada pelo parâmetro do método. Caso não existam, a biblioteca criará todas as interfaces e classes e as adicionará no pacote *br.ita.semantic.owl* e *br.ita.semantic.owl.impl* respectivamente. Posteriormente, o método carregará em memória a ontologia existente no arquivo passado como parâmetro e o inserirá em um objeto *JenaOWLModel*.

Carregada a ontologia em memória, passa a ser possível a realização das diversas operações semânticas. As mais básicas delas são as realizadas em indivíduos (adição, remoção e recuperação). Para realizar estas operações deve-se inicialmente instanciar a classe OntoFactory, conforme pode-se ver na figura 3. Nessa mesma figura é apresentado o método *OntoUtilities.persistIntoFileOntology(String fileDestination, JenaOWLModel owlModel)*, que é o responsável por persistir em um arquivo a ontologia.

```
public static void main(String args[]) {
    OntoMapping[] ontoMap = loadOntoMapping();
    // inicializacao da ontologia
    JenaOWLModel owlModel = OntoUtilities
            .initOntologySystem("file:///home/kabart/ontol/new/DefesaCivil.owl");
    // inicializacao da fabrica
    OntoFactory ontoFac = OntoFactory.getInstance(owlModel, ontoMap);
    // criacao de individuos
    DefaultHelicoptero hel16 = (DefaultHelicoptero) ontoFac
            .createNewIndividual("HEL16", "DefaultHelicoptero");
    DefaultHelicoptero hel7 = (DefaultHelicoptero) ontoFac
            .createNewIndividual("HEL17", "DefaultHelicoptero");
    // retrieve de individuos
    DefaultHelicoptero hel2 = (DefaultHelicoptero) ontoFac.getIndividual(
            "HEL05", Helicoptero.class);
    // remocao de um individuo
    ontoFac.removeIndividual("HEL16", hel16);
    // busca de todos os individuos
    Collection<RDFResource> res = ontoFac.getAllIndividuals("DefaultHelicoptero");

    // persist file
    OntoUtilities
            .persistIntoFileOntology("/home/kabart/test.owl", owlModel);

}
```

Figura 3. ITA Semantic Library - Manipulaçào de Indivíduos.

Após entender como realizar as manipulações básicas de uma ontologia, o próximo passo é entender como usar os recursos de reasoning e regras existentes em uma aplicação semântica. Uma Missão é constituída de mais de uma solicitação de apoio. O questionamento que deve-se fazer é: em uma situação limite, como uma catástrofe natural, onde existem mais necessidades do que meios disponíveis como priorizar esses atendimentos, ou seja, que solicitações de apoio devem ser atendidas?

Uma abordagem sintática seria usar um sistema especialista que mapeasse durante a concepção do sistema ou carga do mesmo as regras a serem definidas, com base em critérios pré-definidos, podendo eles serem dinâmicos ou não.

No caso de uma aplicação semântica, uma regra pode ser adicionada ou removida em tempo de execução, sem a necessidade de reinicialização do sistema, tendo os parâmetros que a define construída com um conjunto de valores totalmente aleatório.

Na figura 4 é apresentado um exemplo de uso de regras na aplicação alvo. Note que a regra define o que é uma Vítima com Gravidade Média usando a linguagem SWQRL, de posse do resultado (figura 5), o sistema poderia definir quais as solicitações de apoio devem ser atendidas.

```
JenaOWLModel owlModel =
    OntoUtilities.initOntologySystem("file:///home/kabart/ontol/new/DefesaCivil.owl");
OntoSQWRL swrlEngine = OntoSQWRL.getInstance(owlModel);
SQWRLResult result=
    swrlEngine.getQuerySQWRLResult("GravidadeMediaVitima",
        "esta_localizada_em(?vitima, ?atingido)" +
        " ∧ Estado(?vitima, \"Isolado\") → sqwrl:" +
        "select(?vitima, \"Media\")");
SQWRLResult result2=
    swrlEngine.getQuerySQWRLResult("GravidadeMediaVitima",
        "esta_localizada_em(?vitima, ?atingido)" +
" ∧ Estado(?vitima, \"Morto\") → sqwrl:select(?vitima, \"Media\")");
```

Figura 4. ITA Semantic Library - Exemplo de Manipulação de Regras.

Note que a variável *result2*, redefine a regra GravidadeMediaVitima, fazendo com que o valor anterior associado a regra seja totalmente modificado, tornando o valor de *result2* diferente do existente na variável *result1*. É importante mencionar que se caso a ontologia for persistida, essa regra também o seria, sendo carregada durante a inicialização do sistema.

```
Vitima: http://localhost/onto/DefesaCivil.owl#VITIMA003
Vitima: http://localhost/onto/DefesaCivil.owl#VITIMA001
Vitima: http://localhost/onto/DefesaCivil.owl#VITIMA006
```

Figura 5. ITA Semantic Library - Resultado de Aplicações de Regras.

A última funcionalidade existente na biblioteca em voga é a que possibilita a reclassificação de indivíduos com base em mecanismos de reasoning. Para implementar essa funcionalidade é usada a classe *OntoReasoning* que recebe quatro parâmetros em seu construtor. O primeiro deles é a ontologia em memória. Na sequência é passado o motor de reasoning desejado, podendo ser passado o *Pellet* ou o *OWLAPI*. Posteriormente se define a ativação ou não do mecanismo de autosincrinização, ou seja, se os indivíduos serão classificados automaticamente ao serem criados, sem a necessidade de que o usuário pedir explicitamente essa ação. Finalmente, é passado se o motor de inferência estará ativo. Essa funcionalidade é apresentada na figura 6, o *OntoReasoning* é usado para recuperar as superclasses de Helicóptero.

```
//inicializacao da ontologia
JenaOWLModel owlModel = OntoUtilities.initOntologySystem("file:///home/kabart/" +
        "ontol/new/DefesaCivil.owl");

//inicializacao do reasoning
OntoReasoning reasoning = OntoReasoning.getInstance(owlModel, "Pellet",
        true, true);
//Como usar a maquina de inferencia
Collection inferredSubclasses= reasoning.getInferedClasses(owlModel,
        "SuperClasses", "Helicoptero");
for(Iterator it = inferredSubclasses.iterator(); it.hasNext();) {
OWLNamedClass curClass = (OWLNamedClass) it.next();
System.out.println("\t" + curClass.getName());
}
```

Figura 6. ITA Semantic Library - Aplicações de Reasoning.

Para finalizar, a biblioteca também fornece um meio de consultas usando a linguagem SPARQ. A supracitada

linguagem é um dos mecanismos definidos pela Web Semântica para realizar consultas em bases ontológicas, tendo como vantagem a sua simplicidade quando comparada a linguagens de consultas a banco de dados como o SQL. Em sistemas altamente dinâmicos, como os de defesa, a linguagem SPARQL pode ser usada para realizar consultas não formatadas a diversas ontologias diferentes, sendo um recurso poderoso em buscas em repositórios semânticos (OWL-S). O uso dessa funcionalidade é realizado através do método estático *OntoQueries.runningSPARQL()*, onde são passados como parâmetros a ontologia e uma String com os requisitos da consulta desejada em uma sintaxe SPARQL.

Entendido como usar a ferramenta, visando validar o experimento, o serviço de atendimento a calamidades foi implementado usando o *Protégé* e a *ITA Semantic Library*, avaliados conforme as métricas apresentadas no início da seção e comparados seus resultados, de forma validar a biblioteca desenvolvida.

De posse do cenário de calamidades, três serviços foram desenvolvidos de forma semântica, os relacionados a Solicitação de Apoio. Essa escolha se deve ao fato desses serviços terem uma natureza mais complexa e ser mais dependente de informações semânticas que os demais.

Os serviços de Solicitação de Apoio desenvolvidos foram três: o SA1, SA2 e SA3. O primeiro desses serviços, o SA1, é o que lista todas as solicitações de apoio existentes no sistema. O segundo (SA2) trata de definir uma ordem de prioridade na Solicitação de Apoio com base na gravidade da vítima. Por fim, o último serviço diz respeito a seleção do meio a ser utilizado com base no tipo de meio de acesso.

Para implementar o primeiro, é necessário apenas uma consulta SPARQL a base de informações ontológicas. Já o segundo é necessário a criação de uma regra que defina que as vítimas de gravidade alta são aqueles que estão com o estado Ferido/Doente e estão localizados em um Local Atingido. Por fim, o último é mais complexo, pois além de implementar uma consulta SPARQL para recuperar o meio de acesso ao local atingido (Terrestre, Fluvial ou Terrestre), precisa inferir o meio de transporte a ser utilizado através de uma superclasse que é TransporteAéreo, TransporteFluvial e TransporteTerrestre.

Com base na primeira métrica, Número de Classes Necessárias para a Construção do Serviço Semântico, os três serviços foram implementados segundo as duas abordagens e avaliados, e um resumo dessa avaliação pode ser vista no gráfico 7. Nele pode-se perceber que no primeiro serviço, o SA1, a ITA Semantic Library necessita da instanciação de mais classes, porém quanto mais complexo a atividade semântica, maior o número de classes que é necessário manipular no Protégé, enquanto através da biblioteca supracitada, o valor é praticamente constante.

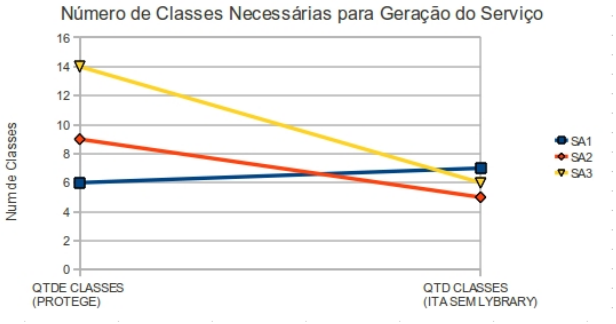

Figura 7. Aplicação de Métrica 01 - Número de Classes Necessárias para representar um Conceito.

Os resultados da aplicação da segunda métrica é apresentada no gráfico 8. Nele é fácil visualizar que o número de linhas de código-fonte necessárias para codificar um conceito usando o Protégé é aproximadamente 100

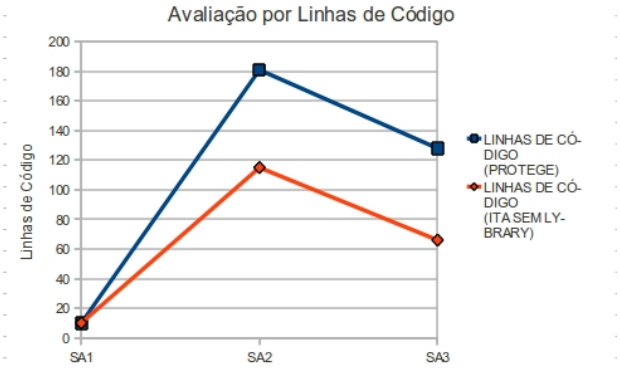

Figura 8. Aplicação de Métrica 02 - Número de Linhas de Código Fonte.

VI. Considerações Finais e Trabalhos Futuros

Conforme pode-se ver no presente documento, para atender as necessidades de difusão e uso de serviços em um ambiente operacional, o projeto dos mesmos usando uma arquitetura orientada a serviço é a mais adequada. Para aumentar a capacidade dos nós clientes em encontrar esses serviços, o uso de informações semânticas para descrever os mesmos é desejável, porém para se obter todos os benefícios do uso de semântica em uma aplicação, é necessário seu uso também na sua implementação.

O uso de semântica em sistemas de defesa tem por finalidade aumentar a capacidade de interoperabilidade dos diversos nós existentes na rede, bem como a construção de sistemas dinâmicos, onde comportamentos não mapeados em tempo de compilação podem ser possíveis de serem usados em tempo de execução.

No presente trabalho foi apresentada uma biblioteca que tem por finalidade simplificar a construção de aplicações semânticas. Como validação da mesma foram utilizadas duas métricas: número de classes e número de linhas de código-fonte, onde a biblioteca foi superior nas duas métricas quando comparada a sua correlata, o Protégé.

Apesar da biblioteca endereçar os aspectos principais acerca de uma aplicação semântica, aspectos específicos de serviços semânticos ainda não foram abordados, como por exemplo métodos que implementem mecanismos de buscas de serviços. Uma outra limitação da biblioteca é o uso de mecanismos de regras proprietários, limitando o uso da mesma.

Como continuação da pesquisa, deve-se realizar novos experimentos da bibliotecas em novos cenários e um estudo de viabilidade para a sua incorporação no framework apresentado em [4]. Além disso, deve-se experimentar e estender a mesma para usar mecanismos de regras não proprietários, viabilizando que a biblioteca possa ser usada nos mais diversos contextos.

REFERÊNCIAS

[1] Alberts, D.S.; Garstka, J.J. e Stein, F.P. Network Centric Warfare.CCRP, USA, 2004.

[2] Denning, D.E. Information Warfare and Security, Addison-Wesley, December 1998.

[3] Office of the Chief of Naval operations, "Naval Command and Control", Naval Doctrine Publication 6, Departmente of the Navy, Ejército USA, 1995.

[4] Marques, H.C.; Parente, J.M.P. e Costa, P.C.G. C2 Framework for interoperability among an air component command and multi-agencies systems. XV ICCRTS, 2010.

[5] Isotani, S.; Mizoguchi, R.; Bittencourt, I.I. e Costa, E. Estado da Arte em Web Semântica e Web 2.0: Potencialidades e Tendências da Nova Geração de Ambientes de Ensino na Internet. Revista Brasileira de Informática na Educação, Volume 17, Número 1, 2009.

[6] Sheth, A. et all.Semantic Association Identification and Knowledge Discovery for National Security Applications. Technical Memorandum 03-009, LSDIS Lab, Computer Science, the University of Georgia, August 15, 2003. Prepared for Special Issue of JOURNAL OF DATABASE MANAGEMENT on Database Technology for Enhancing National Security, Ed. Lina Zhou. (Invited paper, subject to revision).

[7] Saadati, S. e Denker, G. An OWL-S Editor Tutorial - Version 1.1. SRI International Menlo Park, CA 94025. Disponível em: http://owlseditor.semwebcentral.org/documents/tutorial.pdf. Acesso em: 05/06/10.

[8] Bell, M. Introduction to Service-Oriented Modeling. Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley & Sons, 2008.

[9] Bell, M. SOA Modeling Patterns for Service-Oriented Discovery and Analysis. Wiley & Sons, 2010.

[10] Fielding, R. Irvine,U.C., Gettys, J., Mogul, J. Frystyk, H. Masinter,L. Leach,P. e Berners-Lee, T. RFC2616 - Hypertext Transfer Protocol - HTTP/1.1. June 1999.

[11] Bray,T., Paoli,J., Sperberg-McQueen, C. M., Maler, E. E Yergeau, F. Extensible Markup Language (XML) 1.0 (Fifth Edition). November 2008.

[12] Yu, L. Introduction to the Semantic Web and Semantic Web Services. Chapman & Hall/CRC, 2007.

[13] W3C. Web Services Description Language (WSDL) Version 2.0. W3C Member Submition. Acessado em: http://www.w3.org (21/04/2010).

[14] Xinyang Feng; Jianjing Shen; Ying Fan. REST: An alternative to RPC for Web services architecture. First International Conference on Future Information Networks, 2009 (ICFIN 2009).

[15] W3C. SOAP Version 1.2. W3C Member Submition. Acessado em: http://www.w3.org (21/04/2010).

[16] OASIS. Universal Description, Discovery and Integration v3.0.2 (UDDI). OASIS. Acessado em: http://www.oasis-open.org/specs/ (21/04/2010).

[17] Antoniou, G. e Harmelen, F. A Semantic Web Primer. The MIT Press, Cambridge, 2008.

[18] W3C. OWL-S: Semantic Markup for Web Services . W3C Member Submition. Acessado em: http://www.w3.org(21/04/2010).

[19] Lara,R., Polleres,A., Lausen,H., Roman, D., Bruijn,J. e Fensel, D. A Conceptual Comparison between WSMO and OWL-S. DERI, 2004.

[20] Feickert, A. e Lucas, N.J.. Army Future Combat System (FCS) (Spin-Outs) and Ground Combat Vehicle (GCV): Background and Issues for Congress. Congress Research Service, 2009.

[21] Joahsen, F.T.; Hafsoe, T. e Skjegstad, M. Web Services and Service Discovery in Military Networks. 14th ICCRTS, 2009.

[22] Silva, V.S., Silva,J.C.P. e Campos, M.L. Alinhamento de Ontologias através de Algoritmos de Alinhamento Local de Caminhos. Seminário de Pesquisa em Ontologia do Brasil, 2008.

[23] BRASIL. DCA 100-22 - Serviço de Gerenciamento de Fluxo de Tráfego Aéreo. Comando da Aeronáutica, 2010. Acessed: http://publicacoes.decea.gov.br/index.cfm.

[24] W3C Semantic Web Activity News - SPARQL is a Recommendation. W3.org. 2008-01-15. http://www.w3.org/blog/SW/2008/01/15/. Retrieved 2009-10-01.

[25] Jena - A Semantic Web Framework for Java. Disponível em: http://jena.sourceforge.net/. Acessado em: 23/06/10.

[26] Protégé Ontology Editor and Knowledge-base Framework. Disponível em: http://protege.stanford.edu/. Acessado em: 23/06/10.

[27] Erich Gamma,Ralph Johnson , Richard Helm,John M. Vlissides . Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional; 1 edition (November 10, 1994).

[28] Connor, M.A. e Das, A. SQWRL: a Query Language for OWL. Stanford Center for Biomedical Informatics Research. Acessado em: 19/07/2010. Disponível em: http://bmir.stanford.edu.

[29] Horrocks,I at al. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004. Acessado em: 19/07/2010. Disponível em: http://www.w3.org/Submission/SWRL/.

[30] Fentan, N. and Pfleeger, S.L. Software Metrics: A Rigorous and Practical Approach, Revised. Course Technology; 2 edition (February 24, 1998).

[31] Força Aérea Brasileira. Acessado em: 19/07/2010. Disponível em: www.fab.mil.br.

[32] Exército Brasileiro. Acessado em: 19/07/2010. Disponível em: www.exercito.gov.br.

[33] Marinha do Brasil. Acessado em: 19/07/2010. Disponível em: www.mar.mil.br.