FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
MESTRADO EM INFORMÁTICA APLICADA

Carlos Gustavo Oliveira Fernandes

XPLAINS: UMA ABORDAGEM PARA GERAÇÃO DE EXPLICAÇÕES DE FLUXOS DE SERVIÇOS WEB SEMÂNTICOS

FORTALEZA - CE
2008

FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
MESTRADO EM INFORMÁTICA APLICADA

Carlos Gustavo Oliveira Fernandes

XPLAINS: UMA ABORDAGEM PARA GERAÇÃO DE EXPLICAÇÕES DE FLUXOS DE SERVIÇOS WEB SEMÂNTICOS

Dissertação apresentada ao Curso de Mestrado em Informática Aplicada da Universidade de Fortaleza como requisito parcial para a obtenção do Título de Mestre em Informática.

Orientador: Prof. Dr. João José Vasco Peixoto Furtado

FORTALEZA - CE
2008

F363x Fernandes, Carlos Gustavo Oliveira.
Xplains: uma abordagem para geração de explicações de fluxos de serviços web semânticos / Carlos Gustavo Oliveira Fernandes. - 2008.
117 f.

Cópia de computador.
Dissertação (mestrado) – Universidade de Fortaleza, 2008.
"Orientação : Prof. Dr. João José Vasco Peixoto Furtado."

1 Web semântica. 2. Representação de conhecimento. 3. Internet. 4. Inteligência artificial. I. Título.

CDU 681.3:004.822

CARLOS GUSTAVO OLIVEIRA FERNANDES

XPLAINS: UMA ABORDAGEM PARA GERAÇÃO DE EXPLICAÇÕES DE FLUXOS DE SERVIÇOS WEB SEMÂNTICOS

Data de aprovação: _____ / _____ / 2008.

Banca examinadora:

_____

Prof. Dr. João José Vasco Peixoto Furtado
Prof. Orientador - Universidade de Fortaleza - UNIFOR

_____

Prof. Dr. Paulo Pinheiro da Silva
Membro - The University of Texas at El Paso - UTEP

_____

Prof. Dr. Pedro Porfírio Muniz Farias

Membro - Universidade de Fortaleza - UNIFOR

Ao Senhor, aos meus pais,
às minhas irmãs e
à minha querida noiva.

“The only way to have a friend is to be one”.

Ralph Waldo Emerson

Agradecimentos

Aos meus pais, Anderson e Rita; minhas irmãs, Virgínia e Carol; e meu cunhado André (e seus pais) por tudo que me ensinaram, pela valiosa amizade que tenho com todos eles e pelos bons momentos que desfrutamos em família.

A minha amiga e noiva Juliana, que mesmo sendo tão atarefada quanto eu, estava a qualquer hora disponível para me ajudar com seu lindo sorriso. É impossível mensurar o quanto ela é importante para mim, mas como um apaixonado, o que tenho a dizer é que lhe amo.

A minha segunda família, Tarcísio, Júlia, Tatiana, Renata e Consola, que me acolheram como um filho e irmão durante todos esses anos. Sou muito grato a vocês.

Ao meu orientador, Vasco Furtado, pela paciência, motivação e confiança em mim depositada. Agradeço também por todos os ensinamentos repassados e pelos "puxões de orelha" quando eu parecia ter perdido o "senso de urgência". Só não foi melhor, porque como rubro-negro tive que diariamente repetir o nome do rival "vasco", mas valeu a pena. Obrigado professor por nos ensinar a viver Pesquisa.

A todos os colegas que passaram pela célula Engenharia do Conhecimento – UNIFOR, em especial aos amigos Adriano, Daniel e Leonardo pelo apoio e convivência comigo durante todo este período. Agradeço também ao amigo Eurico, que foi de grande importância na fase final do meu trabalho. Obrigado pelas valiosas contribuições.

Aos colegas do BNB e ex-colegas do DETRAN no qual convivi durante este período do mestrado.

Aos professores Paulo Pinheiro e Pedro Porfírio, por aceitarem fazer parte desta banca.

A todos que participaram dos testes de avaliação, pela disponibilidade em realizá-la e pelas valiosas contribuições.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo aporte financeiro.

Aos meus amigos que de todas as formas contribuíram na conclusão do meu trabalho, em especial: Rafael, Mateus, Anderson, Igor, Heitor, Bia, Lia, Mabel e Maria Cristina.

Resumo da Dissertação apresentada ao Corpo Docente do Curso de Mestrado em Informática Aplicada da Universidade de Fortaleza como parte dos requisitos necessários para a obtenção do título de Mestre em Informática (M.Sc.)

XPLAINS: UMA ABORDAGEM PARA GERAÇÃO DE EXPLICAÇÕES DE FLUXOS DE SERVIÇOS WEB SEMÂNTICOS

Carlos Gustavo Oliveira Fernandes

Julho / 2008

Orientador: João José Vasco Peixoto Furtado, D.Sc.

Com a evolução da Web Semântica, o processo de realizar consultas na Web torna-se cada vez mais um processo interativo entre uma pessoa e um agente inteligente. Para responder a essas consultas os agentes devem buscar serviços disponíveis (SWS) na Web e se necessário fazer uma composição entre esses serviços para conseguir o resultado para a consulta do usuário. Além do resultado da consulta, os usuários precisam entender como a solução foi produzida antes que possam confiar e depender dessa solução. Eles precisam de explicações que descrevam o fluxo de SWS e os passos de inferência que foram seguidos para produzir um resultado particular. Neste trabalho definimos um sistema chamado *XPlainS* que automaticamente gera uma infra-estrutura para prover explicações de serviços Web semânticos descritos em OWL-S e que é livre de domínio. Nossa abordagem envolve uma estratégia para geração de regras usadas para representar explicações de execução de SWS. Mostramos a factibilidade e as vantagens de nossa abordagem pela comparação de explicações geradas por *XPlainS* com explicações geradas por sistemas que requerem conhecimento extensivo de domínio, e através de uma avaliação realizada por meio de testes da ferramenta com usuários.

Palavras-chave: Inteligência Artificial, Representação do Conhecimento, Web Semântica, Serviços Web, Serviços Web Semânticos, Explicação.

Abstract of the dissertation presented to the board of faculties of the Master Program in Applied Informatics at the University of Fortaleza, as partial fulfillment of the requirements for the degree of Master of Computer Science (M.Sc.)

XPLAINS: AN APPROACH FOR GENERATING EXPLANATION OF SEMANTIC WEB SERVICES FLOW

Carlos Gustavo Oliveira Fernandes

July / 2008

Advisor: João José Vasco Peixoto Furtado, D.Sc.

With the advent of the Semantic Web, the process of answering user queries on the Web requires more and more interaction between a person and an intelligent agent. To answer these queries, agents must search for semantic Web services (SWS) available on the Web and if necessary, compose them to get a result for the user. Besides the result of their queries, users need to understand how the solution was produced before they can trust and depend on these solutions. They need explanations that describe the SWS flow and the inference steps performed to produce a specific result. For this work, we present a system called *XPlainS* that automatically generates an infra-structure to provide explanations of SWS described in OWL-S, which is domain free. Our approach involves a strategy to generate rules used to represent explanations of SWS execution. We prove the feasibility and present the advantages of our approach by comparing explanations generated by *XPlainS* with explanations generated by systems that require extensive knowledge of the domain, and through an evaluation performed through tests applied to users.

Keywords: Artificial Intelligence, Knowledge Representation, Semantic Web, Web Services, Semantic Web Services, Explanation.

SUMÁRIO

Lista de figuras

Lista de tabelas

Abreviaturas

Sigla	Significado
API	**Application Programming Interface**
B2B	**Business to Business**
BDI	**Belief Desire Intention**
BPEL	**Business Process Execution Language**
BPEL4WS	**Business Process Execution Language for Web Services**
BPWS4J	**Business Process Execution Language for Web Services Java**
DAML	**DARPA Agent Markup Language**
DAML + OIL	**DARPA Agent Markup Language + Ontology Inference Layer**
DAML-S	**DARPA Agent Markup Language Services**
IRI	**Internationalized Resource Identifier**
IW	**Inference Web**
JTP	**Java Theorem Prover**
OWL	**Ontology Web Language**
OWL-QL	**OWL Query Language**
OWL-S	**Ontology Web Language for Services**
PML	**Proof Markup Language**
RDF	**Resource Description Framework**
SBC	**Sistemas Baseados em Conhecimento**
SDS	**Semantic Discovery Service**
SOAP	**Simple Object Access Protocol**
SWS	**Serviços Web semânticos**
UDDI	**Universal Description, Discovery and Integration**
URI	**Uniform Resource Identifier**
W3C	**The World Wide Web Consortium**

Sigla	Significado
WSDL	**Web Service Description Language**
WSMO	**Web Service Modeling Ontology**
XML	**EXtensible Markup Language**

1. Introdução

1.1 Motivação

A Web surgiu com o objetivo de facilitar o compartilhamento de conteúdo por meio de sites estáticos. Com a popularização da web, as grandes organizações identificaram um novo nicho de mercado na área de e-commerce, ou seja, comércio eletrônico, e começaram a investir nessa área. Esses investimentos deram início a uma nova fase da Web, que mudou sensivelmente a forma de interação do usuário com os sites, que passaram a ser dinâmicos.

Recentemente, a Web passou por um novo ciclo de mudanças, chamada de Web 2.0, onde os usuários passaram de consumidores de informação para serem também geradores de informação, através do compartilhamento de informações em sites de relacionamento, *blogs* e *sites* colaborativos (Wikipédia[1], Wikicrimes[2], etc.).

O volume de informação na Web cresceu de forma significativa, mas também de forma desordenada. Para máquinas, um dos grandes problemas da Web atual é o desconhecimento da semântica de seu conteúdo. A fim de usarem

[1] Wikipedia, a enciclopédia livre - Para mais informações acesse: http://www.wikipedia.org

[2] Wikicrimes, mapeando crimes colaborativamente – Para mais informações acesse: http://www.wikicrimes.org

informação na Web, os agentes de software necessitam que o significado dessa informação seja conhecido. Essa anotação semântica inicia uma nova fase da Web, chamada de Web Semântica. Nessa fase, as informações e serviços da Web poderão ter significados tanto para as pessoas (como ocorre atualmente), como para aplicações de software, permitindo assim a obtenção de novos resultados para as consultas realizadas por usuários na Web.

Neste contexto, consultas feitas por usuários em páginas Web tendem a evoluir em um processo interativo entre uma pessoa e um agente de software inteligente (Berners-Lee, Hendler e Lassila 2001). Baseado nas requisições e preferências dos usuários, agentes inteligentes devem ser capazes de encontrar serviços e, se necessário, realizar uma composição automática desses serviços para produzir os resultados esperados pelos usuários.

Este processo de automação da composição de serviços tem contribuído para a concepção de 'organizações virtuais', onde organizações são dinamicamente geradas e conectadas em resposta a um objetivo específico (ou vários objetivos). Serviços Web são componentes proeminentes nesse contexto, pois permitem a interoperabilidade entre sistemas. Isto faz com que seu uso se torne cada vez mais freqüente, tanto por sites que desejem disponibilizar serviços, como por organizações que desejem implantar soluções *business to business* (B2B). Esta evolução da Web, que a torna mais rica semanticamente, faz com que serviços Web ganhem novas possibilidades em aplicações (Hendler, T e Miller 2002).

Serviços Web semânticos (SWS) são serviços que possuem descrições formais de suas funcionalidades, de seus parâmetros de entrada e saída, e essas descrições são usadas para prover procedimentos de descoberta automática, composição e execução (Mcilraith, Sont e Zeng 2001).

À medida que esses processos automáticos avançam, um aspecto ainda mais importante deve ser considerado: usuários de SWS têm a necessidade de entender não apenas resultados, mas também a solução utilizada para que os usuários possam confiar e usar esses resultados. Eles precisam de explicações que

descrevam o fluxo de SWS e os passos que foram seguidos para produzir um resultado particular.

Outro aspecto relevante para explicação de fluxos em SWS é a influência da composição automática e distribuição dos componentes no processo de explicação, pois a dinamicidade da composição dos serviços não permite um conhecimento prévio de como a solução se comportará para produzir o resultado requerido. Desta forma, as explicações desses fluxos devem ser genéricas para quaisquer tipos de solução que utilize serviços Web semânticos.

Muitos autores previamente propuseram métodos para automatizar ou semi-automatizar os processos de descoberta, composição e execução de serviços Web (Mcllraith, Son e Zeng 2001) (M. Paolucci, et al. 2002). Entretanto, essas propostas geralmente não endereçam o aspecto fundamental de usabilidade da necessidade de explicação, apesar de algumas terem começado o processo de fazer serviços explicáveis em nível limitado (McGuinness, Mandell, et al. 2005).

1.2 Objetivo e proposta

Para que usuários possam entender um resultado apresentado por suas consultas, eles devem ser capazes de perceber como este resultado foi produzido. Nesta dissertação, propomos um *framework* que seja capaz de prover esse tipo de explicação ao usuário. Essa explicação deve se basear em informações que permitam ao usuário compreender o fluxo de processo e as decisões tomadas dentro deste fluxo que possam ter causado impacto no resultado de sua consulta.

Esse *framework*, chamado *XPlainS*, permite a geração automática de explicações de fluxos de execução de SWS. Esta abordagem envolve uma estratégia para geração de regras que representam os passos de execução dos SWS. Tais regras são representadas em *Proof Markup Language* (PML) (Pinheiro, McGuiness e Fikes 2004), uma ontologia usada como base de uma interlíngua para

o *framework* de explicação *Inference* Web (IW) (McGuinness e Pinheiro da Silva 2004).

Para apresentação da explicação ao usuário, foi desenvolvida uma ferramenta que utiliza nossa abordagem de representação e gravação dos fluxos de execução de SWS. Esta ferramenta possibilita ao usuário manipular os registros de execução dos serviços através de uma interface de perguntas e respostas sobre como o fluxo de execução foi processado, quais serviços foram invocados, quais falharam, o porquê da execução de um processo e não de outro, ordem de execução, etc. Com isso, esta interface permite ao usuário assimilar informações que possibilitem a compreensão de como o resultado de sua solicitação foi encontrado.

Acreditamos que nossa maior contribuição é prover uma abordagem de explicação que visa suprir as seguintes necessidades do usuário:

- Compreensão de resultados não esperados para suas consultas;
- Entendimento do ciclo de execução de processos.

Além disso, almejamos também uma contribuição em nível de engenharia de software, pois a partir desta solução permitimos que alguém que deseje incorporar explicação a seus serviços Web o faça utilizando este framework. Para isso, algumas pré-condições, como a utilização de uma linguagem que descreva fluxos em serviços Web, devem ser atendidas. Possibilitamos também a depuração de SWS devido à característica de propagação de nossa prova que é capaz de identificar falhas que podem ocorrer em ambientes remotos.

1.3 Organização

Este trabalho está organizado em seis capítulos, incluindo esta introdução. Segue uma pequena descrição de cada capítulo:

O capítulo 2, **Estado da Arte**, apresentará os principais conceitos relacionados com o objetivo deste trabalho e necessários para o entendimento dessa proposta. A seção 2.1 definirá Serviços Web e como eles são utilizados para facilitar a interação entre diferentes aplicações. A seção 2.2 discorrerá sobre o conceito e a arquitetura de Web Semântica, uma nova geração da Web. Na subseção 2.2.1 será definido o conceito de ontologias, o objetivo em se utilizá-las e o padrão OWL. A subseção 2.2.2 e 2.2.3 descreverão Serviços Web Semântico e a especificação OWL-S como um padrão de ontologias para a Web Semântica, respectivamente. A subseção 2.2.4 apresentará como é feita a descoberta e composição automática desses serviços e na sub-seção 2.2.5 apresentará uma ferramenta de descoberta e composição de serviços chamada de SDS. Em seguida, a seção 2.3 apresentará a necessidade de explicação do fluxo de execução dos serviços Web e como o *framework* IW é utilizado como ferramenta de apoio à geração de explicações. Por fim, a seção 2.4 mostrará os trabalhos relacionados com o assunto desse trabalho são descritos, a fim de identificarmos a problemática para esse trabalho.

O capítulo 3, ***XPlainS*: Um framework para explicação de SWS**, apresentará a proposta de uma infra-estrutura de geração de explicação do fluxo de execução de serviços Web semânticos.

O capítulo 4, **A ferramenta de visualização de explicações**, mostrará como as explicações dos fluxos dos serviços são apresentadas em *XPlainS* por meio de sua interface de explicação.

O capítulo 5, **Avaliação de *XPLAINS***, descreverá a pesquisa qualitativa aplicada a usuários para atestar sobre a validade da ferramenta *XPlainS* como infra-estrutura de explicação da execução de processos na Web.

O capítulo 6, **Conclusão**, apresentará a conclusão deste trabalho, bem como os trabalhos futuros identificados.

O apêndice A, **Documentos utilizados na aplicação dos testes**, apresentará uma cópia de todos os documentos (termo de consentimento,

apresentação do cenário de testes e questionário de avaliação) fornecidos ao usuário para a aplicação dos testes da avaliação da ferramenta *XPlainS*.

O apêndice B, **Regras**, apresentará as regras utilizadas para representar os passos de execução do fluxo de SWS.

O apêndice C, **Documentação das classes de *XPlainS***, apresentará a documentação do código-fonte do framework *XPlainS* gerado através do Javadoc[3].

[3] Javadoc Tool - Para mais informações acesse: http://java.sun.com/j2se/javadoc/

2. Estado da Arte

Este capítulo apresenta os conceitos relevantes para o entendimento da proposta desse trabalho. Primeiramente, o capítulo mostra o conceito de serviços Web e como eles são utilizados para facilitar a interação entre diferentes aplicações. Em seguida, o conceito e a arquitetura de Web Semântica, uma nova geração da Web, juntamente com o conceito de ontologias, suas especificações e objetivos são apresentados. O capítulo descreve também como a Web Semântica utiliza-se de ontologias para facilitar a interpretação de seu conteúdo e como é feita a descoberta e composição automática de serviços Web. Por fim, o capítulo identifica a necessidade de explicação do fluxo de execução dos serviços Web e como o framework Inference Web é utilizado como ferramenta de apoio à geração de explicações. Alguns trabalhos relacionados a este são descritos para que a problemática que serviu de motivação para o desenvolvimento desse trabalho seja identificada.

2.1 Serviços Web

W3C [4] (2004) define serviço Web como uma forma padronizada de interação entre aplicações de diferentes plataformas e linguagens. Esta forma

[4] W3C - World Wide Web Consortium. Para mais informações acesse: http://www.w3.org/

padronizada de iteração é feita através da utilização de um protocolo comum que permite a interoperabilidade entre serviços Web.

Esta interoperabilidade é suportada, pois toda e qualquer comunicação feita entre serviços Web ocorre através de mensagens SOAP (*Simple Object Access Protocol*) (Gudgin, et al. 2007). O protocolo SOAP é baseado inteiramente no formato XML (*EXtensible Markup Language***)** (Bray, et al. 2006), permitindo assim uma maior independência de plataformas e linguagens. Roy e Ramanujan (2001) definem SOAP como sendo um protocolo leve para troca de informação descentralizada em ambiente distribuído. A estrutura da mensagem SOAP é composta de quatro elementos:

- Envelope: descreve o conteúdo da mensagem e como ela deve ser processada (obrigatório);
- *Header* (Cabeçalho): contém os dados do cabeçalho (opcional);
- *Body* (corpo): codificação de uma chamada a um método com seus argumentos de entrada ou uma resposta de uma chamada de um método (obrigatório).
- *Fault* (falha): descreve os erros ocorridos no envio da mensagem (apenas nas mensagens de retorno do servidor).

Para que as aplicações possam interagir entre si, é necessário que cada uma conheça a interface de chamada uma das outras. Por isso, cada serviço Web publica um arquivo de descrição chamado de WSDL (Web *Services Description Language*) (Christensen, Curbera, et al. 2001). WSDL é um arquivo no formato XML que descreve o conjunto de operações disponibilizadas por um serviço Web e como acessá-los. Christensen et al. (2001) conceituam WSDL como sendo um documento que define serviços através da utilização dos seguintes elementos:

- *Types* (tipos): definição do tipo de dados em XML;
- *Messages* (Mensagens): definição dos tipos de dados que serão trafegados durante a troca de mensagens (envio e recebimento);
- *Port Type* (tipo de porta): resumo das operações suportadas;
- *Binding* (ligação): definição do formato das mensagens.

Com o avanço dos serviços Web, as organizações sentiram a necessidade de compartilhar informações com seus clientes e parceiros sobre seus produtos e serviços. Para suprir essa necessidade, surgiu a idéia do repositório UDDI (*Universal Description, Discovery and Integration*) (Sperberg-McQueen 2003), que permite que aplicações publiquem, acessem e descubram serviços na Web.

A arquitetura de serviços Web é dividida em três componentes: provedor do serviço, repositório de serviços e cliente que requisita/acessa o serviço. Esses três elementos interagem através das seguintes ações: publicação, pesquisa e invocação dos serviços.

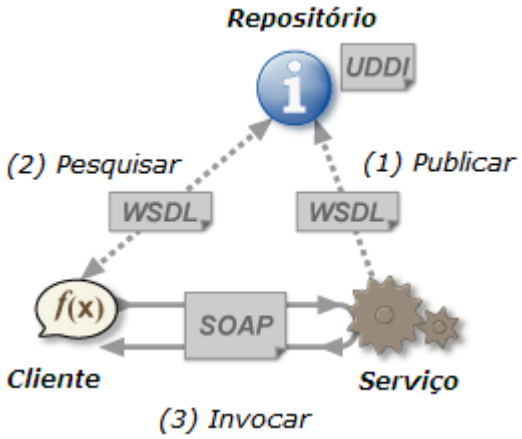

Figura 1 Arquitetura de serviços Web. Fonte: Figura adaptada de Voormann (2006)

A Figura 1 ilustra bem essas interações. O número 1 mostra que o serviço ao ser criado dever registrar-se em um repositório de serviços chamado UDDI. Para realizar este registro, o provedor disponibiliza algumas informações a respeito do serviço (dados do provedor, finalidade, etc.) e um arquivo no formato XML chamado WSDL, que contém as descrições operacionais do serviço. O número 2 mostra que um cliente que deseje utilizar um tipo serviço deve realizar uma busca no repositório. Ao encontrar o serviço desejado pelo cliente, o repositório disponibiliza o WSDL para o cliente. De posse do arquivo WSDL, o cliente consegue realizar invocações diretamente ao serviço Web, conforme ilustrado no número 3.

2.2 Web Semântica

Com o constante crescimento da Web, o volume de informações gerado e disponibilizado aumenta a cada dia. O processo de realizar buscas que retornem informações relevantes ao usuário vem se tornando uma tarefa cada vez mais difícil. Isto ocorre devido a não estruturação das informações presentes na Web. A Web Semântica surgiu como uma evolução da Web atual com o objetivo de anotar estas informações com meta-informação que seja compreensível por aplicações de software. Desta forma, os usuários humanos fazem uso das informações e as aplicações de software fazem uso das metas-informações.

Para Berners-Lee, Hendler e Lassila (2001), a Web Semântica é uma extensão da Web atual com significado bem definido para seu conteúdo, facilitando o trabalho cooperativo entre pessoas e máquinas.

Na Figura 2 podemos observar um mapa conceitual que agrupa os componentes e tecnologias que integram o modelo da Web Semântica. Partindo do conceito de XML, identificamos que este possibilita a estruturação de taxonomias e a criação de mapa de tópicos. Por sua vez, as taxonomias são interpretadas como tesauros (por pessoas) ou mapas de tópicos (por máquinas) e esses se combinam para formar ontologias que permitem agentes de software criar redes sociais e trabalhar com aplicações, utilizando de assinaturas digitais. As ontologias também permitem que os serviços Web se comuniquem com aplicações ou busquem metadados que possam ser representados em RDF (*Resource Description Framework*) (Beckett e McBride 2003). O RDF serve de base para a Web Semântica, que por sua vez é uma extensão da Web atual.

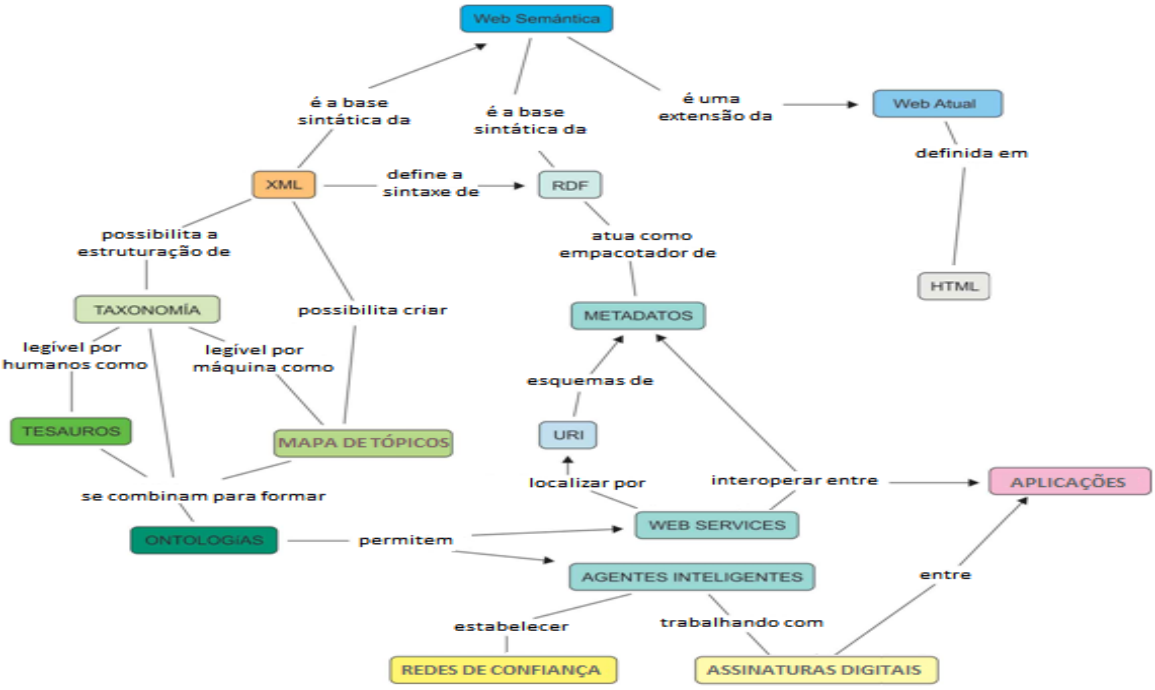

Figura 2 Mapa conceitual da Web Semântica. Fonte: Figura adaptada de Perojo e León (2005)

É importante lembrar que todo o desenvolvimento da Web Semântica ocorre em colaboração entre W3C, pesquisadores e parceiros da indústria.

A arquitetura da Web Semântica pode ser apresentada através da Figura 3. Ela representa uma visão da Web Semântica como sendo formada por um conjunto de camadas interligadas e cada uma dessas camadas acrescenta funcionalidades e expressividade à camada inferior. Essas camadas compreendem as tecnologias necessárias para que o conteúdo da Web possa ser compreendido e usado por computadores e pessoas.

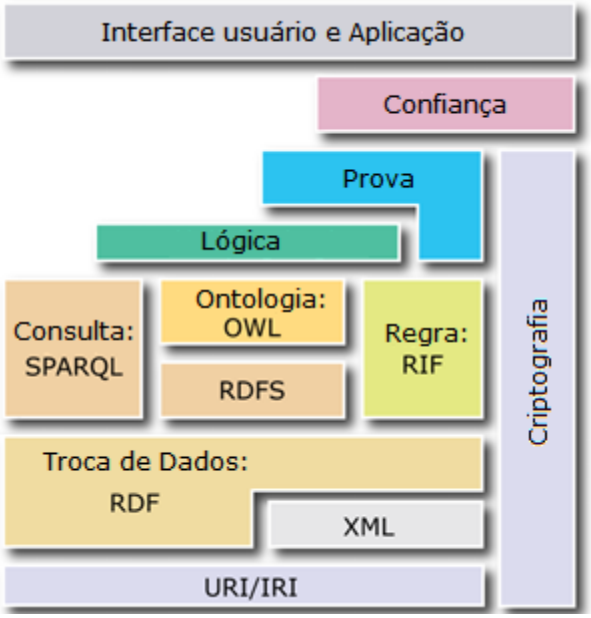

Figura 3 Arquitetura da Web Semântica. Fonte: Figura adaptada de W3C (2007)

Começando pela base da figura, podemos observar a camada URI/IRI que representa os identificadores para os recursos disponíveis na Web. Um identificador é um endereço global único para o recurso. Os recursos na Web podem ser páginas ou sites, documentos pessoais, aplicações, etc.

As próximas camadas representam as linguagens XML e RDF. Essas linguagens são responsáveis pelo encapsulamento, formatação e modelagem dos documentos e dados da Web. A camada de XML encapsula documentos Web utilizando *tags* criadas pelos usuários. A camada RDF descreve os recursos da Web e possibilita que o conhecimento sobre esses dados seja representado, utilizando expressões lógicas. As camadas de linguagens servem de base para as camadas de ontologia e de lógica, e juntas tentam prover semântica para os dados disponíveis na Web. A camada de ontologias funciona como um dicionário, fornecendo um vocabulário comum e com semântica bem definida, necessário para que os termos das informações e as relações entre eles possam ser interpretados corretamente. Dentre as linguagens de ontologias com capacidade de representar a semântica das informações na Web, podemos destacar DAML+OIL.

A camada de lógica possibilita que as informações sejam representadas em um formalismo, para que sejam processadas por raciocinadores automáticos, facilitando assim a construção de inferências. A partir do conhecimento representado nas ontologias, os raciocinadores podem inferir novas conclusões. Temos ainda a camada de prova que é a responsável por justificar o conteúdo das informações trocadas na Web.

Acima de todas essas camadas, temos a camada de Confiança que é responsável por gerar recomendações de confiança para o conteúdo da Web e da autoria das transações previstas pelas aplicações. Através de certificados digitais e outros mecanismos, a camada de confiança tenta garantir que as informações apresentadas sejam confiáveis.

Essa arquitetura permite que aplicações heterogêneas e distribuídas possam compartilhar não só informações, mas também um significado comum para essas informações.

2.2.1 Ontologias

Ontologia, segundo Aurélio (2006), é parte da filosofia que trata da natureza do ser. Em ciência da computação/informação, ontologia é uma representação de primitivas (conjuntos de conceitos, relações entre os conceitos e atributos) utilizadas para modelar o conhecimento de um domínio (T. R. Gruber 2008).

O uso de ontologias é comum entre pessoas, organizações e sistemas de software, com o intuito de unificar os termos de um domínio, facilitando assim a comunicação e correta compreensão das informações pelas partes envolvidas. Para Gruber (1995), uma ontologia especifica conceitos e tem por propósito permitir o compartilhamento e reuso de conhecimento. Para Heflin (2004), uma ontologia compreende o conhecimento de um domínio específico, tornando-o assim reusável.

As ontologias têm um papel importante na Web Semântica, pois possibilitam que a semântica seja utilizada por aplicações Web e agentes inteligentes. A Web Semântica requer ontologias estruturadas em nível de classes de um domínio de interesse, relações entre as classes e suas propriedades. Ainda neste contexto, estas são utilizadas para especificar conceitos compartilhados entre sistemas, comércio eletrônico, gerência de conhecimento, prover serviços de consultas e facilitar a interoperabilidade entre sistemas heterogêneos e distribuídos.

Atualmente existem várias formalismos e ferramentas para criação e manipulação de ontologias. A padronização de Web Semântica da W3C sugere o formalismo OWL para desenvolvimento de ontologias.

Ontology Web *Language*, OWL, foi uma especificação criada para evoluir DAML+OIL e criar um padrão de ontologias para serem utilizadas na Web Semântica. McGuinness e van Harmelen (2004) definem OWL como sendo uma linguagem utilizada por aplicações que necessitam processar o conteúdo de informações e não apenas apresentá-las ao usuário. Essa linguagem facilita a interpretação das informações disponíveis na Web. OWL provê maior capacidade de interpretação de informações, por acrescer semântica às informações, diferente de outras linguagens como: XML, RDF e RDF Schema (Brickley, Guha e McBride 2004).

2.2.2 Serviços Web Semânticos

Um dos principais objetivos da Web Semântica é permitir que os agentes de software possam compreender e, portanto utilizar as informações disponibilizadas na web. Porém, a estrutura de um dos componentes da Web, os serviços Web, não suporta esta idéia. Alesso e Smith (2005) afirmam que essas tarefas não são totalmente realizáveis com as tecnologias atuais, pois é necessário primeiro criar uma linguagem adequada que permita essa automatização. Isto se dá pelo fato de arquivos que descrevem os serviços Web, o WSDL, não permitirem aos agentes

descobri-los e utilizá-los sem a interferência humana, pois essas especificações não incluem significado ou semântica das operações e mensagens lá descritas.

Por serem importantes componentes nessa nova Web, a integração com a Web Semântica vem sendo uma área bastante explorada pelos pesquisadores do mundo. Essas pesquisas visam definir uma forma de descrever semanticamente os serviços Web que permita aos agentes de software descobrir, selecionar e executar os serviços Web de forma automática.

Algumas pesquisas apontam para prováveis propostas que poderão ser adotadas como padrão de anotação semântica de serviços Web, como por exemplo: OWL-S, SWMO (Roman, et al. 2005), SAWSDL (Farrell e Lausen 2006), WSDL-S (Akkiraju, et al. 2005), METEOR-S (Rajasekaran, et al. 2004). Na próxima seção apresentaremos OWL-S, que juntamente com SWMO são apontadas como tecnologias candidatas para o desenvolvimento de SWS.

2.2.3 OWL-S

Martin, Paolucci et al. (2005) desenvolveram uma especificação chamada OWL-S que descreve a função das operações de serviços Web (precondições e efeitos) e os tipos semânticos para cada uma das entradas e saídas. OWL-S, anteriormente chamada DAML-S (PAGELS 2000), é uma ontologia OWL composta por três sub-ontologias: ontologia de serviço (*Profile*), ontologia de processo (*Process Model*) e ontologia que detalha os protocolos de transporte (*Grounding*) (Martin, Paolucci, et al. 2005).

A Figura 4 apresenta a função de cada uma dessas sub-ontologias utilizadas na representação do serviço Web. Nela podemos observar que a ontologia *Profile* apresenta a funcionalidade do serviço, enquanto *Process Model* descreve como ele funciona e por fim *Grounding* informa como o acesso ao serviço pode ser feito.

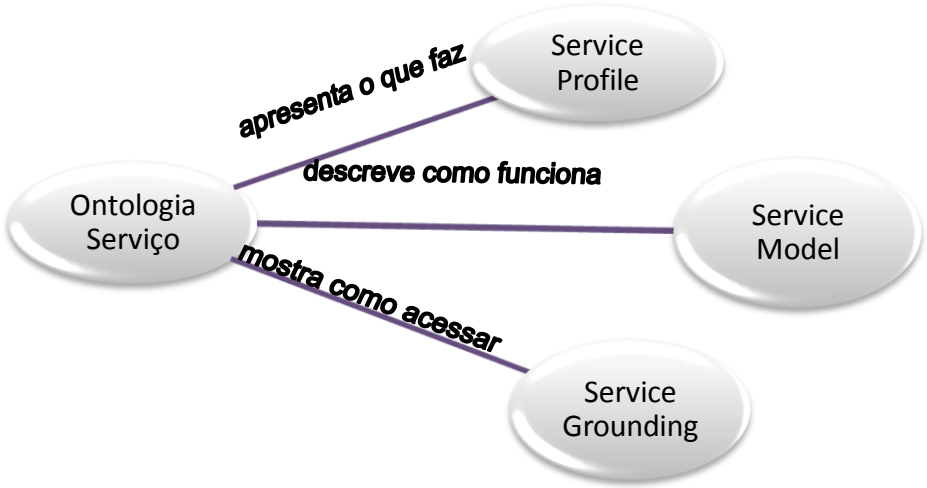

Figura 4 Estrutura OWL-S. Fonte: Martin, Paolucci, et al. (2004)

A ontologia *Profile* tem por objetivo descrever a ontologia de serviços utilizando três tipos de informação: organização que provê o serviço, a finalidade do serviço e as características especiais do serviço. As informações do provedor do serviço simplesmente se referem à entidade responsável por prover o serviço e essas por sua vez referenciam informações de contato do responsável por executar o serviço. As informações que descrevem a finalidade do serviço, em outras palavras, “o que o serviço faz”, especificam as entradas requeridas e a saída gerada. Portanto, se o serviço apresenta alguma condição para ser executado ou efeitos que alterem essas condições, a ontologia *Profile* é responsável por descrevê-las. A Figura 5 apresenta uma instância da ontologia *Profile*. Na linha 2 podemos observar a que serviço ela se refere, na linha 3 o nome deste serviço e na linha 4 e 5 sua entrada e saída, respectivamente.

```
1 <profile:Profile rdf:ID="FindBookProfile">
2         <service:isPresentedBy rdf:resource="#FindBookService"/>
3         <profile:serviceName xml:lang="en">Book Finder</profile:serviceName>
4         <profile:hasInput rdf:resource="#BookName"/>
5         <profile:hasOutput rdf:resource="#BookInfo"/>
6 </profile:Profile>
```

Figura 5 Instância da ontologia *Service Profile* para busca de livro

Para descrevermos como é feita a interação com um serviço Web, podemos visualizá-lo como sendo um processo. A ontologia *Process Model* descreve como este processo funciona. Um processo não representa o serviço a ser executado e sim uma especificação de como interagir com o serviço. Segundo Martin, Paolucci et al. (2004) a ontologia de processo é composta por:

- Participantes: são os envolvidos no processo;
- Entradas e saídas: informações requeridas para execução do processo e informações retornadas ao solicitante;
- Pré-condições e resultados: condições que devem ser atendidas para execução do processo e efeitos produzidos pela execução do processo;
- Saídas condicionais e efeitos: condições para ocorrências do resultado.

Os processos podem ser classificados como simples, atômicos ou compostos.

Processos simples não podem ser invocados, pois não são associados a uma ontologia *Grounding*. Processos simples são utilizados para prover uma interface para o processo ou uma representação simplificada para processos compostos.

Um processo atômico representa um acesso direto a um serviço Web. Ele representa um serviço a ser executado com uma única interação com o solicitante do serviço, através de uma chamada e de um único retorno.

Por outro lado, processos compostos compreendem um conjunto de outros processos, que por sua vez podem ser atômicos ou compostos. Processos compostos podem requerer um ou mais interações entre o solicitante e os serviços que estão sendo executados. Um processo composto mantém o estado de execução e à medida que os serviços trocam mensagens o processo avança. Para isso o processo utiliza-se de estrutura de controles definidas, como por exemplo:

- Sequence: conjunto de processos que devem ser executados na ordem definida;
- Split: conjunto de processos que devem ser executados concorrentemente. Essa estrutura é finalizada assim que todos os processos sejam programados para execução;
- *Split Join:* conjunto de processos que devem ser executados concorrentemente. Essa estrutura é finalizada assim que todos os processos concluam sua execução;
- *Any-Order:* conjunto de processos que são executados sem uma ordem prévia. Essa estrutura é finalizada assim que todos os processos concluam sua execução;
- Choice: um processo é selecionado para executar dentre um conjunto de outros processos.
- If-then-else: estrutura de controle que verifica uma condição para executar ou não o(s) processo(s) associado(s).
- Iterate: estrutura de controle abstrata que executa iterações em uma quantidade indefinida de vezes. É uma classe abstrata que serve como superclasse para repeat-while e repeat-until.
- Repeat-While e Repeat-Until: estrutura de controle que realiza iterações dos processos até que uma condição seja atendida.

A ontologia *Grounding* define a forma de acesso a um serviço Web. Ela detalha protocolos, formatos de mensagens, serializações, transporte e endereçamento. Essa ontologia realiza um mapeamento entre os conceitos descritos nas entradas e saídas dos processos, com as entradas e saídas do serviço Web.

Enquanto as ontologias *Profile* e *Process Model* são representações abstratas, a ontologia *Grounding* pode ser vista como uma representação concreta, pois está associada a um serviço específico.

2.2.4 Descoberta e composição automática de serviços Web

A Web Semântica através da criação de ontologias que descrevam os serviços Web, espera prover as seguintes funcionalidades:

- **Descoberta automática de serviços:** automatização do processo de localização de serviços de acordo com as necessidades do cliente. Os serviços Web podem ser localizados através dos serviços de buscas baseados em ontologias ou através de registro em repositórios públicos;
- **Invocação automática:** através da descoberta automática dos serviços Web, permite que agentes de software realizem sua execução automatizada;
- **Interoperabilidade e composição:** composição de serviços Web utilizada para realização de tarefas complexas, baseada nas entradas, saídas, precondições e efeitos. Este processo é bastante importante, principalmente quando a solicitação de um usuário não pode ser realizada com a execução de apenas um serviço. Desta forma, os agentes de software buscam compor soluções que possam atender a requisição do usuário.

Neste contexto, diversas propostas surgem indicando como atingir essas metas, pois além do uso das ontologias para descrever os serviços, faz-se necessário desenvolver aplicações que consigam buscar e compor os serviços dinamicamente. A seguir apresentaremos uma proposta chamada SDS (Mandell e McIlraith 2003) (*Semantic Discovery Service*) que foi desenvolvida pelo laboratório KSL (*Knowledge Systems Laboratory*) da Universidade de Stanford.

2.2.5 *Semantic Discovery Service*

Semantic Discovery Service (SDS) é a união de uma linguagem de especificação de processos em serviços Web (*Business Process Execution Language for* Web *Services* - BPEL4WS) (OASIS 2000) com OWL-S e raciocinadores da Web Semântica (Mandell e McIlraith 2003).

A linguagem BPEL4WS é utilizada para coordenar as interações entre serviços Web. Ela permite a composição manual de serviços através de modelagem de processos. Entretanto, por falta de tipos de dados apropriados e relacionamentos de classes, BPEL4WS não permite a descoberta e integração automática de serviços Web.

Para suprir essa limitação, SDS funciona como um *proxy* entre BPEL e serviços Web. A interação entre BPEL4WS e SDS ocorre através de restrições funcionais e não-funcionais (definidas pelo usuário) em OWL-S ou pela invocação de parâmetros para os serviços.

SDS utiliza uma linguagem de consulta, chamada OWL-QL (*OWL Query Language*) (Fikes, Hayes e Horrocks 2003) para realizar consultas na base de conhecimento das descrições dos serviços (OWL-S) para encontrar serviços. As consultas são suportadas por um servidor OWL-QL e executadas pelo raciocinador JTP (KSL *Java Theorem Prover*) (Richard, Jenkins e Frank 2003). Caso não seja encontrado um serviço que atenda aos parâmetros de entradas e/ou saída

requeridos, o SDS se encarrega de compor uma solução que se adeqüe aos parâmetros dentre os serviços disponíveis.

A Figura 6 permite visualizar o fluxo de interação entre as tecnologias envolvidas na descoberta/ composição automática de SDS. No item um, BPWS4J (IBM 2002) (plataforma de criação e execução de BPEL4WS) envia uma mensagem de invocação para SDS contendo as restrições dos serviços em OWL-S. Ao receber esta mensagem, no item dois, SDS separa as restrições de serviços em uma consulta OWL-QL e invoca o servidor OWL-QL. No item três, o servidor OWL-QL usa o JTP para pesquisar serviços descritos em OWL-S que atendam a consulta. No item quatro, o servidor OWL-QL retorna as descrições dos serviços que atenderam a consulta. No item cinco, SDS registra provas de explicação no framework IW. No item seis, caso a consulta realizada pelo OWL-QL não tenha encontrado nenhum serviço que atenda as restrições de entrada e saída, o SDS, através de seus serviços parceiros, busca compor uma solução que resolva essas restrições. No passo sete, a composição é retornada para o SDS e este retorna o fluxo e a prova gravada para o BPWS4J como mostrado no item oito.

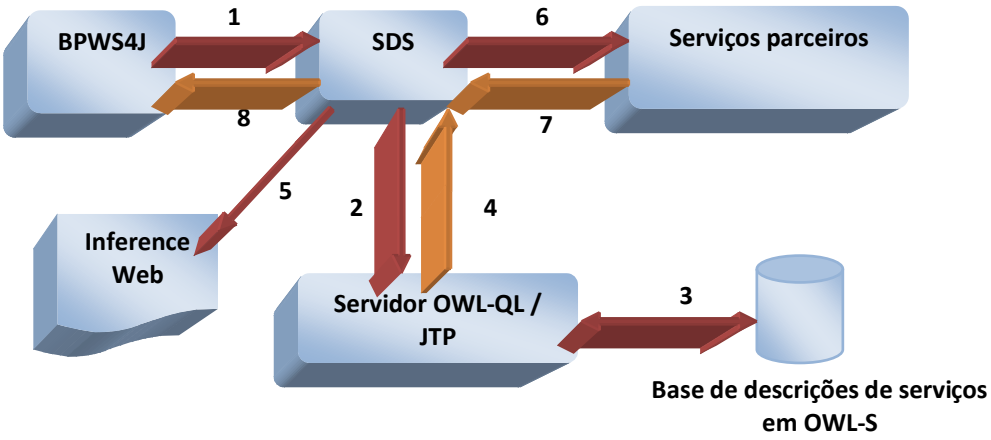

Figura 6 Fluxo de interação de SDS

Uma característica importante em SDS, é que ele possui uma integração com o framework de explicação IW, permitindo a geração de explicações para

soluções que dê maior confiabilidade às tarefas de composição automática de serviços Web e disponibilize uma infra-estrutura para auditorias. Na próxima sessão iremos falar mais sobre o framework IW.

2.3 Realização de explicações na Web

Aplicações heterogêneas e distribuídas fazem uso dos serviços Web para capturarem as informações necessárias para responder as consultas dos usuários. Em ambientes heterogêneos como a Web, as fontes das informações podem estar distribuídas em diversos locais. O resultado final das consultas de um usuário é resultado da integração das diversas consultas e buscas de serviços realizadas. Com isso, esse resultado pode não ser compreendido pelo usuário que pode querer uma explicação para entender como esse resultado foi obtido.

Inference Web (IW) é um *framework* que possibilita às aplicações gerarem explicações portáveis e distribuídas para suas respostas (McGuinness e Pinheiro da Silva 2004). Através de IW, os usuários podem obter as fontes das informações, detalhes sobre os raciocinadores e as linguagens utilizadas para geração de explicações. O IW atende as necessidades de aplicações que utilizam raciocinadores automáticos para geração de suas respostas em ambientes como a Web. As informações fornecidas por IW tentam dar maior credibilidade às respostas apresentadas pelos raciocinadores.

Para atender a necessidade de conhecimento da fonte das informações obtidas, IW provê dados como: nomes das fontes, data e autor da informação, certificado de autoria, grau de confiança e grau de completude da informação. IW também provê informações sobre o rastro do raciocínio executado por uma aplicação ou uma composição de aplicações. Tais informações podem não ser apropriadas para usuários comuns da web e por isso IW pode transformar este rastro de raciocínio em explicações mais adequadas aos usuários. Com o aumento de aplicações que utilizam soluções distribuídas pela Web, as fontes de informações

tendem a ser distribuídas e faz-se necessária a utilização de uma linguagem portável, para que as provas possam ser compartilhadas entre sistemas. IW trata deste requisito de portabilidade de provas com a utilização da linguagem PML (McGuinness e Pinheiro da Silva 2004), uma linguagem baseada em OWL, utilizada para representar as provas e informações sobre elas.

Um conjunto de ferramentas é disponibilizado pelo framework IW para manipulação das provas. Para exemplificar o uso dessas ferramentas, utilizaremos o exemplo do domínio (McGuinness, Hsu, et al. 2003). Neste exemplo um agente de software busca encontrar a melhor combinação de vinho para a comida que é especialidade de Tony. O resultado obtido com essa consulta é que a especialidade de Tony é frutos do mar (*SEAFOOD*).

- ***IW Registrar****:* ferramenta de manipulação das meta-informações utilizadas na geração de provas. A Figura 7 exemplifica a ilustração das informações da ontologia "Especialidade de Tony" armazenada no *IW Registrar*.

Figura 7 Detalhes da ontologia em *IW Registrar*

- ***IW Explainer:*** ferramenta para prover uma explicação para consultas realizadas pelo usuário em formato mais amigável e mais

compreensível. A Figura 8 exibe a explicação do resultado da consulta: “Qual especialidade de Tony?”.

Figura 8 Visualização da explicação do resultado em *IW Explainer*

- ***IW Browser:*** ferramenta para apresentação de provas. As provas podem ser visualizadas em formato de árvore ou em sentenças na língua inglesa. A Figura 9 exibe a árvore de provas para a consulta de especialidade de Tony.

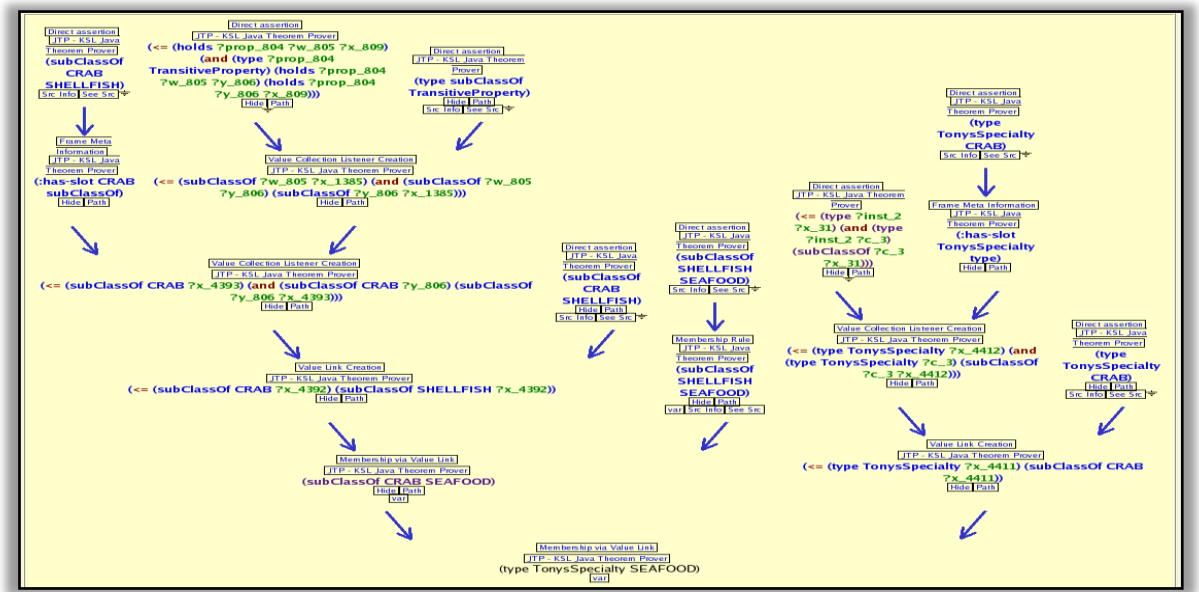

Figura 9 Visualização da prova em *IW Browser*

2.3.1 PML

A linguagem PML desenvolvida para representação de provas apresenta dois componentes principais: os passos de inferência (*Inference Steps*) e os nós das provas (*nodesets*) (McGuinness e Pinheiro da Silva 2004). Um *nodeset* representa um nó da prova e contém as informações referentes à sua conclusão, aos nós a partir dos quais a conclusão foi derivada, nós antecedentes, e a regra de inferência aplicada para obter a conclusão. Os *nodesets* apresentam também informações sobre a linguagem utilizada para representação das informações. Dessa forma, uma prova pode ser vista como um conjunto de *notesets* ligados por passos de inferência. Os passos de inferência representam as regras de inferência que foram aplicadas para que a conclusão do *nodeset* fosse derivada.

A Figura 10 apresenta um trecho de uma prova em PML. As informações de um *NodeSet* estão apresentadas nas linhas de número seis a dezessete na

figura. A linha sete representa a conclusão do *nodeset*. A linha oito traz a informação da linguagem utilizada na prova. As informações sobre o *InferenceStep*, ou passo de inferência aplicado para concluir esse *nodeset,* estão apresentadas nas linhas de número dez a quinze. Sobre esse passo de inferência, a prova traz as informações sobre a máquina de inferência utilizada (linha onze), regra de inferência (linha treze) e quais são os antecedentes da nó derivado (linha catorze).

```
1<rdf:RDF
2 xmlns:iw="http://www.inference-Web.org/2004/07/iw.owl#"
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
4<iw:Language rdf:about="http://www.inference-Web.org/registry/LG/English.owl#English"/>
5<iw:DeclarativeRule rdf:about="http://www.inference-Web.org/registry/DPR/GMP.owl#GMP"/>
6<iw:NodeSet rdf:about="http://localhost/owls/proof.owl#nodeproof0_0">
7   <iw:hasConclusion>RESULT(Amazon).</iw:hasConclusion>
8   <iw:hasLanguage rdf:resource="http://www.inference-Web.org/registry/LG/English.owl#English"/>
9   <iw:isConsequentOf>
10     <iw:InferenceStep>
11       <iw:hasInferenceEngine rdf:resource="http://www.inference-
Web.org/registry/IE/SWIPL.owl#SWIPL"
12         rdf:type="http://www.inference-Web.org/2004/07/iw.owl#InferenceEngine"/>
13       <iw:hasRule rdf:resource="http://www.inference-Web.org/registry/DPR/Hyp-Resolution.owl#Hyp-
Resolution"/>
14    <iw:hasAntecedent  rdf:resource="http:// localhost/owls/proof.owl#proof1_0"/>
15       </iw:InferenceStep>
16   </iw:isConsequentOf>
17</iw:NodeSet>
18</rdf:RDF>
```

Figura 10 Pedaço de prova PML do exemplo Especialidade de Tony

2.4 Trabalhos relacionados

Explicações para Sistemas Baseados em Conhecimento (SBC) tornaram-se um tópico significante e independente de estudo. O projeto NEOMYCIN (Clancey 1986) contribuiu para a pesquisa de explicação em SBCs, usando uma representação explícita para estratégias de resolução de problemas e metas-regra

para planejamento de explicações. Entretanto, foi específico para MYCIN (Buchanan e Shortliffe 1984) em termos de resolução de problemas e representação de domínio.

WOZ (Shankar, TU e Musen 1998) é um framework para explicação de sistemas de suporte a decisão baseados em componentes. O framework é composto de componentes funcionais que representam o processo de raciocínio, os agentes de visualização cooperativos associados que são responsáveis por apresentar a explicação e a interação com o usuário, e os modelos de domínio da aplicação, tais como modelos de usuário, modelos de agentes e estratégia de explicação. Apesar do fato de WOZ não ser limitado a um sistema em particular, ele não é facilmente escalável, já que uma nova estratégia de explicação deve ser desenvolvida para cada aplicação.

As ferramentas de depuração de Comércio Eletrônico (Lieberman e Wagner 2003) contribuíram para a pesquisa de explicação de fluxo de execução de processos utilizando um agente que rastreia a interação do usuário com o site de comércio eletrônico. Eles desenvolveram um sistema que fornece explicações para processos de comércio eletrônico, baseados em representação de um modelo de tarefa, e descreve as atividades do processo.

Há também trabalhos de explicação de paradigmas de raciocínio e representações particulares que possam ser usadas para capturar alguns tipos de sistemas baseados em regras e processos, incluindo os trabalhos com objetivo de explicação de lógica descritiva (D. McGuinness 1996).

Explicações para CALO, um agente de software cognitivo, depende da utilização do ambiente de execução de SPARK BDI (Morley 2004) para processamento de tarefa. As explicações resultantes são apresentadas através da categorização de tipos de explicações e estratégias de pergunta/resposta (McGuinness, Glass, et al. 2007). O trabalho de explicação para CALO é geral, tem o objetivo de explicar assistentes cognitivos que aprendem apesar de seu projeto ser em torno de uma arquitetura de agente *Belief Desire Intention* (BDI) para execução de tarefa.

Em (Vaculin e Sycara 2008), uma forma de gravação do rastro de execução dos processos é proposta para monitoração do fluxo de execução de serviços descritos em OWL-S. Pelo rastro de execução, é possível realizar monitoração da execução do serviço, através da depuração do fluxo ou simulação de sua execução. Eles também propõem uma extensão de OWL-S para tratamento de falhas na execução dos processos.

Com a automatização de processos e soluções, a preocupação em fornecer dados aos usuários que permitam a compreensão do processo executado ou a busca por credibilidade dos resultados encontrados ou até mesmo depurações da execução de processos, torna-se cada vez maior.

Todavia, com base nos trabalhos apresentados anteriormente, encontramos restrições no que diz respeito à geração de informação livre de domínio para explicação de fluxos de processos distribuídos e compostos dinamicamente. Em trabalhos como (Clancey 1986), (Shankar, TU e Musen 1998), soluções de explicações aos usuários são apresentadas, porém possuem uma alta vinculação com o domínio, e assim, para cada novo sistema criado com necessidade de prover explicação uma solução teria de ser refeita. Outras soluções como (McGuinness, Glass, et al. 2007), apesar do desacoplamento com o domínio do problema, são dependentes da infra-estrutura de execução, não permitindo assim o seu reuso em outras estruturas de execução. Em trabalhos como (Vaculin e Sycara 2008), apesar de possuírem desacoplamento do domínio do problema e da infra-estrutura de execução, porém apresentam limitações quanto à distribuição dos serviços. O foco principal da solução é o monitoramento dos eventos relevantes que ocorrem durante a execução do processo e não o provimento de explicação para o usuário de como esses eventos ocorreram. Portanto, podemos concluir que as atuais abordagens apresentam limitações em relação à distribuição da solução, criação automática da solução e desacoplamento das arquiteturas.

Nos próximos capítulos, será mostrado como foi concebida nossa proposta que busca preencher as lacunas deixadas pelos trabalhos anteriores. Nossa proposta foi desenvolvida em três etapas: representação dos passos de execução em uma estrutura que permita a extração de explicação, gravação de

traces de execução em processos distribuídos e compostos automaticamente e apresentação da explicação ao usuário.

Para que fosse possível criar um ambiente que permitisse a validação de nossa proposta, tivemos que realizar algumas escolhas de especificações ou ferramentas, como por exemplo: ontologia de serviço, ferramenta para descoberta e composição de serviços Web e especificação para explicação.

A especificação utilizada para descrever os serviços Web, foi OWL-S. Esta escolha foi motivada principalmente pela disponibilidade de uma API (*Application Programming Interface*) desenvolvida em Java (Mindswap 2004), capaz de interpretar e executar serviços Web descritos em OWL-S. Ao utilizar esta API, pudemos diminuir o esforço em programar nossos exemplos e casos de uso. Entretanto, qualquer outra ontologia poderia ter sido escolhida para descrever os serviços Web. Para tanto, precisaríamos desprender um esforço adicional para desenvolver APIs que interpretassem e executassem os serviços semânticos descritos nessas ontologias.

A ferramenta utilizada para descoberta e composição de serviços, foi SDS. Esta ferramenta foi escolhida, pois a mesma apresenta uma excelente integração com OWL-S e por possuir uma característica muito interessante, que é a geração de provas no mesmo formato utilizado neste trabalho, o que vem a enriquecer ainda mais nossa proposta. Com isso, podemos apresentar uma explicação de como foi feita a descoberta e composição dinâmica dos serviços Web. Da mesma forma que escolhemos a ontologia OWL-S para ser utilizada nesse trabalho, a escolha de SDS também foi feita visando facilitar o desenvolvimento deste trabalho. A não utilização desta ferramenta implicaria somente na perda da explicação de busca e composição de serviços que é gerada pela ferramenta. Entretanto, nossa proposta não seria prejudicada, pois o restante da explicação continuaria sendo gerada.

Para representação da explicação, escolhemos o *framework* IW, por apresentar uma estrutura para explicações de provas e ferramentas para manipulação dessas provas.

3. *XPlainS*: Um framework para explicação de SWS

Neste capítulo será apresentado o *framework XPlainS* e como ele foi projetado e desenvolvido.

3.1 Motivação

Conforme apresentado no capítulo anterior, diversos trabalhos propõem métodos de explicação. Porém, geralmente essas metodologias não são livres do domínio, o que as torna inapropriadas para o uso em explicação de serviços Web semânticos. Outras metodologias se propõem a realizar explicações genéricas, mas por outro lado são dependentes da arquitetura da qual executam as tarefas, o que também não é apropriado no contexto de SWS. Nosso trabalho procurou desenvolver uma solução que contemple as restrições apresentadas por estes trabalhos. Para isso, nos focamos em atingir três sub-objetivos: i) definir uma metodologia de representação do fluxo de execução de processo de forma que esta metodologia facilite a geração de explicações, ii) definir uma metodologia para realizar as gravações dos *traces* de execução dos serviços e que se propague

durante a execução dos serviços distribuídos e iii) desenvolver uma ferramenta que acesse os dados e apresente a explicação ao usuário.

3.2 Arquitetura geral de *XPlainS*

O *XPlainS* é um framework desenvolvido com o intuito de prover explicações de SWS. Ele foi dividido em duas estruturas: estrutura de gravação e estrutura de explicação.

A estrutura de gravação é responsável por registrar os *traces* de execução dos serviços Web. Diferentemente de ferramentas de registro de eventos, onde as informações geralmente são gravadas sem estruturação e representação, *XPlainS* utilizada uma forma de representação de conhecimento para cada passo de execução. A partir desta representação podemos gerar as explicações para os usuários.

O processo de gravação dos passos de execução de SWS ocorre em quatro etapas durante a execução dos serviços. Em cada um dessas etapas, dados são coletados e armazenados em nossa estrutura de representação de conhecimento. Essas etapas são:

- Na verificação das condições necessárias para execução do serviço;
- Início da execução de um serviço;
- Finalização do serviço;
- Invocação de serviços Web.

Na Figura 11, podemos observar a arquitetura de *XPlainS*. No lado direito da figura podemos observar a Interface de explicação que deve ser implementada nas ferramentas que interpretam e executam os serviços Web semântico. Na mesma figura podemos observar, também, os quatros pontos principais onde essa interface deve ser referenciada: verificação de condições, inicialização e finalização de

serviços e invocação de outros serviços Web. Em cada um desses pontos ocorre uma interação da interface de explicação com o motor de explicação. A interação de inicialização e finalização de serviços e a invocação a outros serviços Web acessam o motor de explicação para armazenar as regras, capturar as condições que são armazenadas na pilha (através da verificação de condições) e gerar o arquivo PML por meio da utilização de um serviço Web provido por IW *Registrar*.

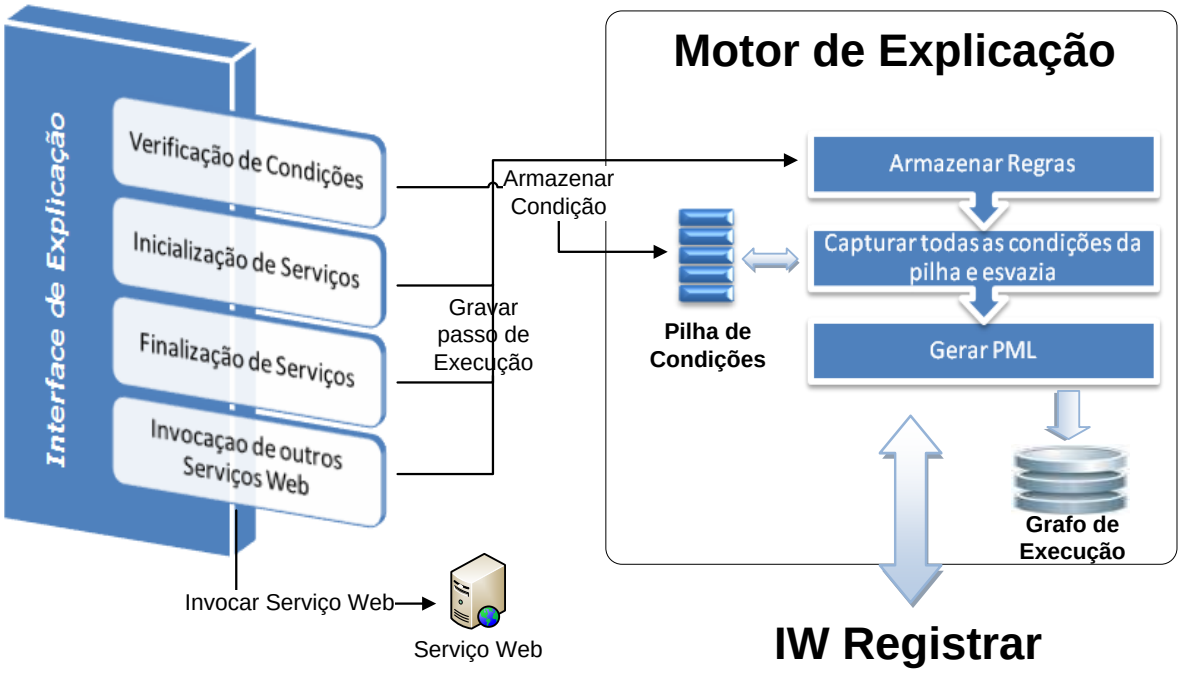

Figura 11 Arquitetura geral de *XPlainS*

A estrutura de explicação provida por *XPlainS* utiliza as informações geradas pela estrutura de gravação e as disponibiliza como explicações em forma de perguntas e respostas, para que auxilie o usuário a compreender o fluxo que produz o resultado de sua solicitação e esta explicação poderá ser utilizada durante, ou após, a execução dos serviços.

No decorrer do capítulo, iremos apresentar em detalhes como foi desenvolvida a representação dos passos de execução que permite a geração de

explicação e o funcionamento da estrutura de gravação de provas, instanciada para ontologia OWL-S.

3.3 Representação do conhecimento

A representação do conhecimento para explicações é inspirada no trabalho de (Glass e McGuinness 2006) no contexto de assistente inteligente, CALO. O rastro do processo de execução é um grafo acíclico onde cada nó do grafo é representado por uma estrutura descrita em PML definida em Inference Web. Como foi descrito em 2.3.1, cada nó em formato PML do grafo de explicação tem três pedaços principais de informação: a regra de inferência (utilizada para guiar a busca por informação), a conclusão (informação declarativa representando a informação concluída pela execução do processo) e os nós antecedentes da regra de inferência. Nossa intenção é criar uma representação declarativa da execução do processo de um serviço semântico a fim de utilizar as ferramentas de manipulação de explicação já desenvolvidas em Inference Web. O conjunto de regras de inferência de processos utilizado neste trabalho é uma adaptação do que foi desenvolvido por Glass e McGuinness (2006). Adicionamos mais cinco regras as regras propostas. Elas são apresentadas na Tabela 1.

A Regra de número 1, por exemplo, declara que uma regra de inferência chamada *Executing* é a manipulação de antecedentes representando que X dá suporte a um objetivo de nível mais alto, as precondições para ativar X foram estabelecidas e X ainda não foi concluído. A conclusão da regra de inferência é que X ainda está em execução. Dessa forma, podemos explicar um questionamento do usuário sobre porque X ainda está sendo executado.

Tabela 1 Regras de Inferência

1	ServiceTopLevelGoal(x) AND IntentionPreconditionsMet(x) AND TerminationConditionsNotMet(x) => Executing(x)
2	TopLevelGoal(y) AND Supports(x, y) => SupportsTopLevelGoal(x)
3	Supports(x, x)
4	ParentOf(y, x) AND Supports (y, z) => Supports (x, z)
5	CompletedPredecessors(x) AND ImplementingProcedureInstance(x, p) AND ProcedurePreconditionsMet(p) => IntentionPreconditionsMet(x)
6	∀y OrderedBefore(y, x) AND Successful (u) => CompletedPredecessor(x)
7	∀c InstancePrecondition(p, c) AND Met(c) => ProcedurePreconditionsMet(p)
8	BottomLevelTask(y) AND Supports(y, x) AND Unfinished (y) => TerminationConditionsNotMet(x)
9	Executing(x) OR Failed(x) OR Future(x) => Unfinished(x)
10	ImplementingProcedureInstance(x, p) AND InstanceTerminationCondition(p, c) AND notMet(c) => Unfinished(d)
11	IsSequence(x) OR IsRepeat(x) OR IsParallel(x) => ImplementingProcedureInstance(x)
12	ServiceTopLevelGoal(x) AND IntentionPreconditionMet(x) AND TerminationConditionMet(x) => Finished(x)
13	BottomLevelTask(y) AND Supports(y, x) AND Finished (y) => TerminationConditionsMet(x)
14	InitializationRepeat(x) AND TerminationRepeat(y) => Repeat(d)
15	∀x Repeat(x) => IsRepeat(x)

Em CALO, o processamento de tarefas é feito no contexto de uma arquitetura de agente BDI, onde existem objetivos e sub-objetivos do agente. No nosso caso, assumimos que o objetivo e sub-objetivo de um processo podem ser extraídos de um fluxo de processos que estão sendo executados. Em outras palavras, o objetivo de maior nível é o processo principal e a hierarquia de sub-objetivos é formada pelos vários sub-processos.

A estrutura de prova definida dá suporte à busca por informação que responde as perguntas selecionadas por um usuário. A busca por esta resposta é guiada pelas regras de inferência. Com o uso dessas regras, é possível agrupar os nós, isto é, definir um conjunto de regras em particular, que sempre que for encontrado na provas, faz referência a um tipo específico de informação. Os três principais tipos de agrupamento são: agrupamento hierárquico, agrupamento de inicialização do processo e agrupamento de finalização do processo. Esses três grupos diferentes estão ilustrados na Figura 12.

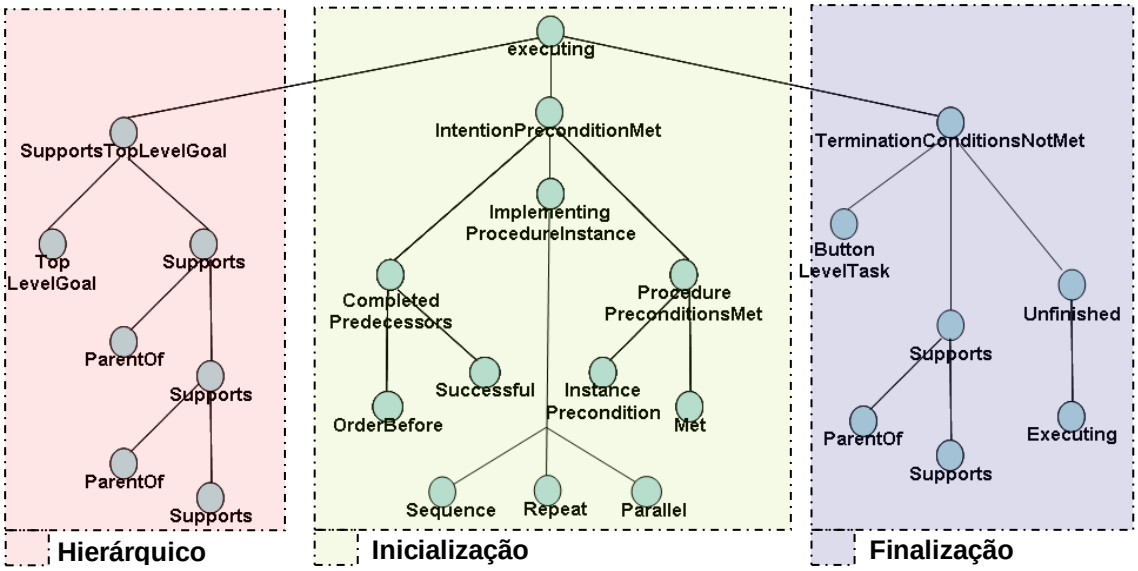

Figura 12 Template do Grafo de Execução

Cada um desses grupos auxiliam a ferramenta de explicação de *XPlainS* a coletar as informações que serão apresentadas ao usuário. Por exemplo, se o usuário deseja obter a informação dos serviços que foram executados até o presente momento, a ferramenta de explicação busca na árvore de provas o agrupamento hierárquico referente ao processo e coleta as informações sobre a hierarquia do processo, formata a explicação (utilizando os *templates* de perguntas/respostas que estão na ontologia) e disponibiliza ao usuário. Caso a informação for o porquê de um serviço A não ter sido executado, então, a ferramenta busca no agrupamento de inicialização a causa da não execução do serviço A, que

neste caso poderia ser porque uma condição não foi atendida. Processo semelhante ocorre quando o usuário deseja saber por que o processo B ainda está em execução. Para isso, a ferramenta acessa o agrupamento de finalização e busca a informação de que o processo A tem um sub-processo B e este ainda encontra-se em execução.

3.4 Geração de explicação

O processo de geração de informação a partir do grafo do fluxo de execução dos serviços é feito em paralelo com a interpretação e execução do fluxo. Como foi dito no Capítulo 2, utilizaremos OWL-S como ontologia de descrição de serviços Web. Utilizamos uma interface de software chamada de OWL-S API (Mindswap 2004) capaz de interpretar esta descrição, tornando-a 'operacional', ou seja, capaz de seguir as estruturas de controle e realizar as invocações dos serviços Web.

O lado esquerdo da Figura 13 mostra o fluxo básico seguido pela OWL-S API. Este fluxo inicia-se pela verificação das precondições do processo a ser executado. Caso as precondições sejam atendidas, o fluxo prossegue com a identificação do tipo de processo que será executado e este pode ser do tipo atômico ou composto. Caso o processo seja do tipo atômico, ele será executado imediatamente. Sendo do tipo composto, o processo precisa ser decomposto nos vários processos que o compõe e o fluxo de execução deve ser iniciado para cada um deles.

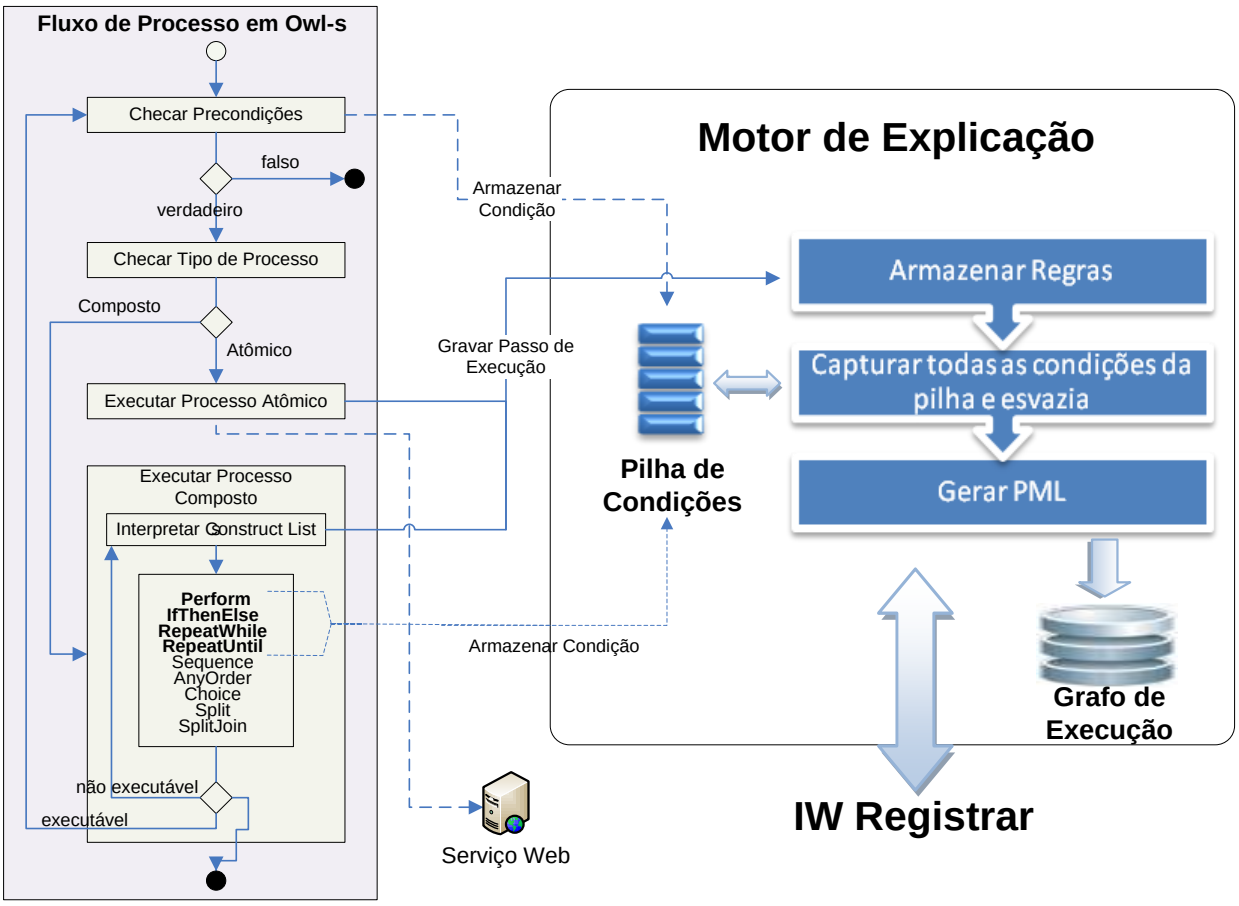

Figura 13 Geração Regras de Execução do processo *XPlainS*

No lado direito da Figura 13, observa-se os componentes de *XPlainS* desenvolvidos para gerar e armazenar os passos de execução da interpretação do fluxo OWL-S. Esta arquitetura é composta por um motor de explicação responsável pelo armazenamento das regras, captura das precondições/condições dos processos armazenados em uma pilha de condições e pela geração de um arquivo de prova em formato PML. A figura apresenta também os pontos onde ocorre a interceptação do fluxo básico de OWL-S API para obter as informações necessárias para geração das provas.

Como dito anteriormente, em OWL-S existem dois tipos de processos que podem ser “executados”, o processo atômico e o composto. O processo atômico é o de fluxo simples que se resume em apenas como será feito o acesso ao serviço Web a ele associado. O processo composto é um conjunto de outros processos

atômicos e/ou compostos. Um processo composto ao ser executado produz um algoritmo recursivo, pois para cada processo composto encontrado, OWL-S API reinicia o fluxo até que todos os processos cheguem a um processo final, geralmente um processo atômico. Note que antes da interpretação do fluxo de processo, uma checagem das precondições e condições (*Repeat*, *IfThenElse etc*) é realizada. Durante esta conferência, o algoritmo de *XPlainS* armazena todas essas condições em uma pilha. Após esta checagem, OWL-S API identifica o tipo de processo que irá executar. No início da execução do processo, a pilha de condições é esvaziada e os nós de provas que representam o processo em execução são armazenados utilizando as regras de processos em execução. No momento de finalização de cada um dos processos, as regras que representavam o processo em execução são atualizadas por regras de processos finalizados. Em resumo, a interceptação do fluxo ocorre em três pontos principais: no momento da interpretação das precondições e condições, e no início e fim da execução dos processos.

3.5 Geração de explicação de serviços distribuídos

Um dos aspectos mais importantes de explicações de serviços Web está no contexto de serviços distribuídos. Suponhamos o seguinte cenário:

> Daniel, um estudante universitário, vai participar de um congresso em Aracaju e deseja comprar um pacote completo de viagem (passagens áreas, hotel, traslados) que seja barato. Para isso, ele acessa o site de sua agência de turismo e informa as datas de sua viagem para que ela retorne o pacote com o menor preço possível. A agência organiza o pacote e retorna toda programação para que Daniel confirme a compra e pague-a. Porém, ele não fica satisfeito com o resultado apresentado, pois discorda do hotel que lhe foi escolhido, um hotel cinco estrelas, que tem um custo bem mais alto que

> outras opções e que, portanto não se enquadra no requisito que o pacote seja barato.

Daniel não tem a mínima idéia do que acontece quando ele fornece suas informações e confirma sua consulta. Não sabe que ao confirmar suas informações, o site da agência de turismo acessa diversos serviços Web como de empresas aéreas, redes de hotéis, empresas de translado etc. e estes por sua vez podem acessar outros serviços Web e assim por diante. No caso de Daniel, o que aconteceu foi que a agência de turismo acessou o serviço da associação hoteleira e este serviço acessa os hotéis e pousadas conveniados para saber a disponibilidade de quartos no período informado. Porém, no instante do acesso muitos dos serviços desses hotéis/pousadas estavam indisponíveis e os que foram acessados estavam com todos os quartos ocupados. Dessa forma, a opção encontrada foi o hotel cinco estrelas de um custo bem mais elevado que os outros que estavam indisponíveis no momento.

Por isso, prover uma explicação que contemple a distribuição da solução é extremamente necessário. Basear-se em uma explicação apenas da chamada aos serviços que são diretamente referenciados não é uma boa opção. Tomemos o exemplo de Daniel. Como ele não se satisfez com o resultado apresentado, provavelmente gostaria de obter mais informações que lhe ajudassem a compreender o porquê do resultado. Em uma explicação baseada somente nas invocações dos serviços que são diretamente referenciados, isso não seria útil, pois no nosso cenário somente teríamos a execução do serviço da associação hoteleira retornando os quartos que foram conseguidos, como o do hotel cinco estrelas, perdendo assim o detalhe de que não foi possível comunicar-se com os outros hotéis/pousadas.

Com a heterogeneidade da Web, processos passíveis de falhas e a dinamicidade fornecida pela composição automática de serviços Web dificultam bastante a preparação a priori da infra-estrutura de explicação. Nossa abordagem resolve este problema ao embutir a geração de um sistema de explicações no nível da ontologia de serviços. Entretanto, é necessário atender alguns requisitos, como:

os serviços Web devem estar descritos em uma ontologia de serviços e utilizando *XPlainS* integrado a API que interpreta o fluxo da ontologia de serviços.

A fim de permitir o registro dos passos de execução dos serviços Web distribuídos e sem requerer mudança na estrutura legada dos serviços Web, adicionamos uma novo componente a *XPlainS* – nomeado gerente de explicação. O gerente de explicação funciona como um *wrapper*, pois ele tem a característica de envolver as chamadas aos serviços Web e também a inicialização da execução de um fluxo de SWS.

O gerente de explicação é automaticamente ativado no momento em que a API invoca um serviço Web. Neste instante, o gerente de explicação realiza duas ações: a primeira é repassar os nós gerados ao serviço *WS_RegistrarProvas* para que ele persista os nós gerados e que estão gravados localmente, e a segunda é anexar a referência do último nó do grafo na chamada do serviço Web para que o próximo serviço dê continuidade à gravação do ponto em que o processo foi requisitado.

É importante lembrar que em soluções distribuídas é comum a ocorrência de falhas, o que no nosso caso ocasionaria a perda das provas que estão sendo gravadas localmente e que seriam repassadas somente ao seu final ou durante a invocação a um novo serviço Web. Por este motivo, o repasse das provas que estão sendo geradas localmente ocorre durante todo o ciclo de execução, evitando assim a perda das provas que estão gravadas localmente em caso de falha. Se o repasse fosse feito somente ao término de toda execução, ocorrendo uma falha, tudo o que tivesse sido gravado localmente seria perdido. Dessa forma, em determinados pontos do ciclo de execução o gerente de provas acessa o serviço *WS_RegistrarProvas* para persistir os passos gravados localmente. Podemos observar essas atividades descritas anteriormente na Figura 14.

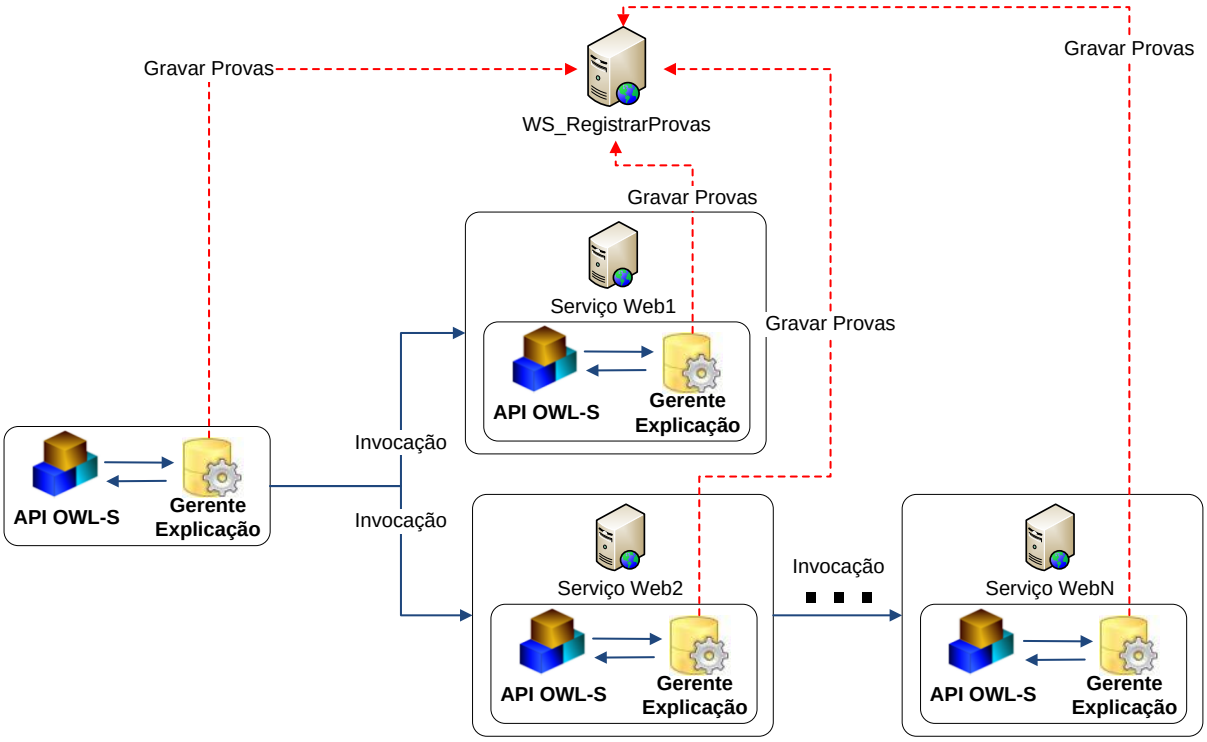

Figura 14 Fluxo de Explicação dos Serviços Web Distribuídos.

Em resumo, o arquivo que contém os passos de execução é gradualmente gerado baseado na execução do fluxo e interação entre os serviços Web e isto nos permite disponibilizar ao usuário uma explicação em tempo real de como o processo está sendo executado, bem como de entender problemas de performance e de indisponibilidade de serviços.

3.6 Integração com as explicações de SDS

Outro aspecto importante em soluções que utilizam SWS são as composições desenvolvidas automaticamente. As ferramentas de descoberta e composição automática de serviços são decisivas nas escolhas dos serviços que

serão executados. Por isso, acreditamos que a capacidade de explicar o que foi executado por essas ferramentas é de extrema importância.

Neste contexto, podemos citar dois tipos de ferramentas que se propõem a fazer descoberta/composição de serviços Web. O primeiro tipo de solução é aquele serviço chamado de “caixa preta”, ou seja, um serviço onde não temos acesso como ele tomou as decisões para buscar o resultado. Outro tipo de solução é a que se propõe a representar como produziu determinado resultado.

Por entendermos que o objetivo principal de nossa proposta é prover ao usuário um entendimento de como o fluxo foi produzido, acreditamos que a explicação de como tais serviços foram selecionados para serem executados é uma característica que provê maior transparência e confiança.

Como já citado no Capítulo 2, no desenvolvimento de nossa solução utilizamos o SDS como ferramenta de busca e composição de serviços. SDS foi escolhido, pois o mesmo possui a capacidade de gerar provas da execução de suas ações no mesmo formato utilizado por *XPlainS*, o formato PML.

Para podermos incorporar a explicação disponibilizada por SDS, foi necessária uma modificação simples na máquina de geração de provas de SDS a fim de criar uma ligação com as duas provas. Esta alteração ocorreu de forma similar à solução que apresentamos para explicação de serviços Web distribuídos. Na Figura 15 apresentamos essa mudança, que foi a adição de um gerente de explicação dentro da máquina de geração de provas de SDS, que também é um serviço Web. O funcionamento é similar ao da solução distribuída, ou seja, no momento em que a ferramenta que executa o fluxo definido na ontologia de serviços chega ao ponto de invocação de SDS, a referência do ultimo nó gerado é anexado para que SDS possa dar continuidade à execução.

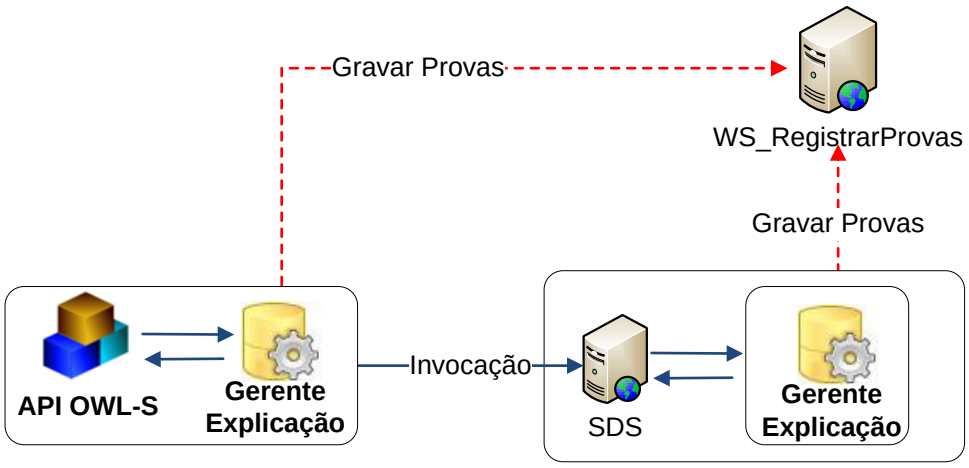

Figura 15 Adaptação em SDS para integrar as gravações das provas

3.7 Conclusão

Neste capítulo, apresentamos o *framework XPlainS* e todos os seus componentes desenvolvidos para prover uma infra-estrutura de explicação da execução de SWS. Detalhamos a arquitetura genérica de *XPlainS*, a representação utilizada para extrair explicações dos rastros de execução de SWS, o processo de geração desses rastros e a integração com a explicação gerada por uma ferramenta de busca e composição de SWS.

Ao final, pudemos atingir dois dos três sub-objetivos (representação dos passos de execução e geração dos passos de execução) propostos e dar suporte para que o último (apresentação das explicações) seja atingindo e mostrado no próximo capítulo.

4. A ferramenta de visualização de explicações

Neste capítulo, apresentamos a ferramenta desenvolvida para apresentar a explicação ao usuário e que nos permite atingir o último sub-objetivo. Esta ferramenta utiliza o fluxo de execução gravado pela arquitetura proposta no capítulo anterior.

4.1 Visualizando as explicações

XPlainS coleta informações sobre a execução do processo para utilizá-las na explicação oferecida ao usuário durante ou após o término dos processos. As explicações seguem uma ontologia que relaciona perguntas e respostas ao momento em que podem ser aplicadas.

Esta relação se faz necessária, pois existem mudanças no diálogo que é apresentado ao usuário quando ele faz uso da ferramenta de explicação durante a execução do processo ou após a execução do processo. As explicações providas por *XPlainS* tem como objetivo principal responder perguntas relacionadas a:

- Proveniência do serviço ou informação;
- Razões que levaram a execução de determinados processos;
- Ordem e tempo de execução dos processos.

Na Tabela 2 podemos observar a relação entre alguns dos tipos de perguntas respondidas por *XPlainS*, com o momento em que a explicação pode ser solicitada. Essas perguntas foram baseadas nas categorizações propostas por McGuinness, Glass et al. (2007).

Tabela 2 Relação Perguntas com o momento da explicação

Tipos de perguntas	Com processo em execução	Com processo concluído
Sobre um processo que ainda está sendo executado	X	
Como processo foi executado		X
Referencia ao tempo de execução		X
Quantidade de vezes que um processo foi executado		X
Sobre o resultado		X
Ordem de execução dos processos	X	X
Sobre o que falta ser executado	X	
Sobre o processo que está sendo executado	X	
Causa de não finalização da execução processo	X	X
Causa de execução de um processo	X	X
Causa da não execução de um processo	X	X
Provedor do serviço	X	X

4.1.1 Interface de explicação

A ferramenta de explicação disponibilizada por *XPlainS* é composta por três componentes visuais: explicativo, histórico de navegação (diálogo travado) e árvore de processos. Cada um destes componentes permite ao usuário navegar de diferentes formas pelas provas. Na Figura 16 podemos observar cada um dos componentes na interface de explicação.

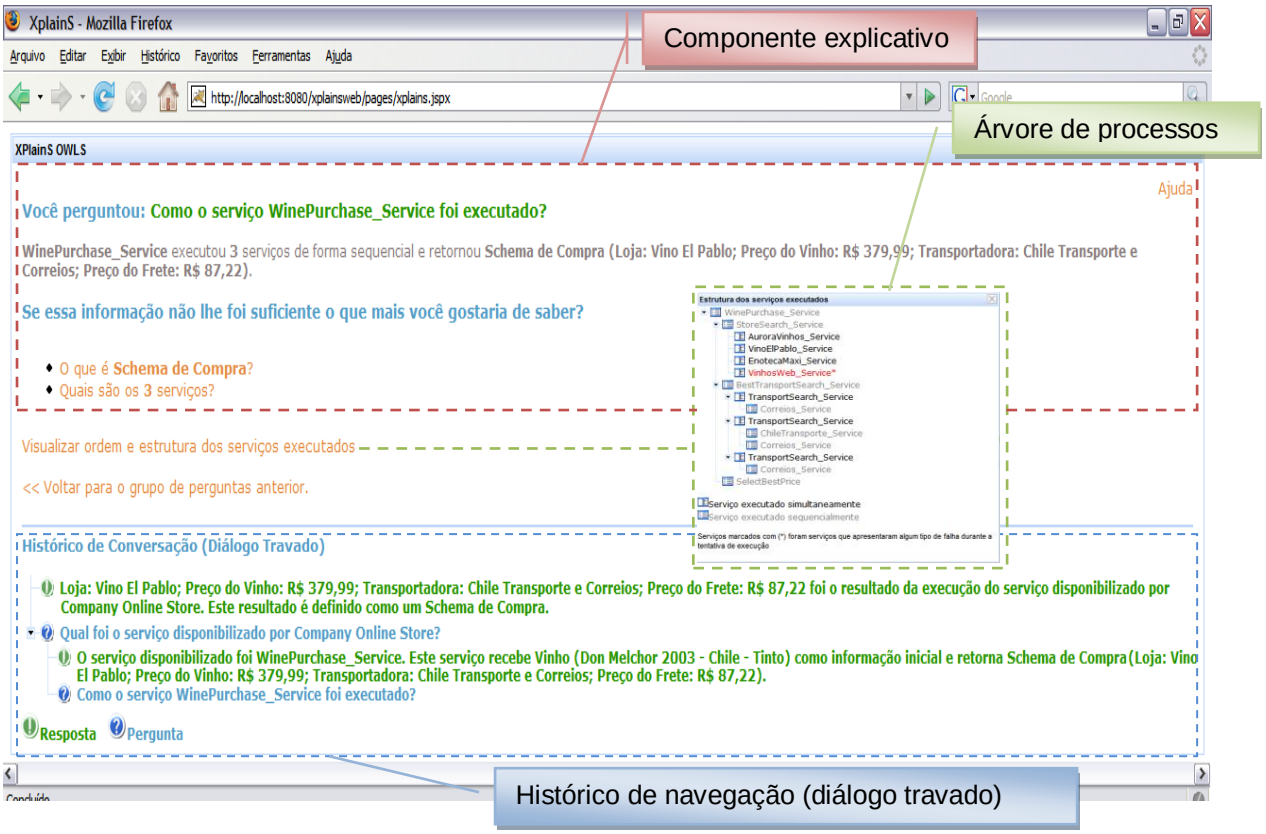

Figura 16 Interface de explicação de *XPlainS*

Para exemplificar o funcionamento de cada um dos componentes da interface vamos trabalhar com o seguinte cenário:

> “Mateus deseja comprar o disco de áudio *Snoopy's Classics Beatles* para dar de presente ao seu filho de cinco anos. Ele então acessa o site Company Online Store, informa que deseja comprar o disco e confirma sua escolha. Após obter a informação da loja onde o disco encontra-se com o menor preço, Mateus sente-se impelido a consultar a ferramenta *XPlainS* para obter maiores explicações do resultado produzido.”

4.1.1.1 Componente Explicativo

O componente explicativo, visualizado na Figura 17, exibe primeiramente a pergunta selecionada pelo usuário. Logo abaixo da pergunta, o componente exibe a resposta para esta pergunta selecionada e oferece um conjunto de outras perguntas para que o usuário possa obter mais detalhes sobre a explicação apresentada.

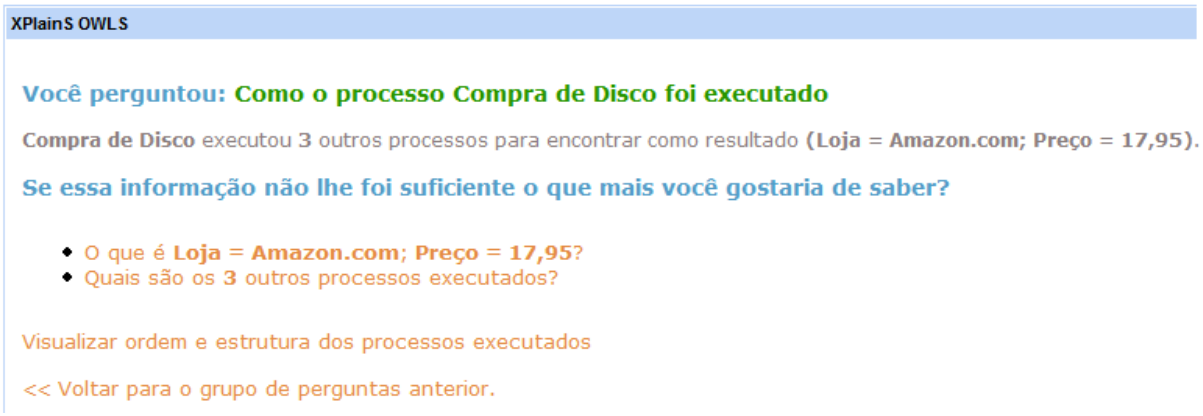

Figura 17 Componente Explicativo

A Figura 17 mostra que o usuário selecionou a pergunta "*Como o processo **Compra de Disco** foi executado?*" e obteve como resposta "***Compra de Disco** executou **3** outros processos para encontrar como resultado **Loja = Amazon.com; Preço = 17,95.***". Abaixo dessa resposta a ferramenta apresenta ao usuário novos questionamentos acerca da resposta apresentada.

Ao selecionar uma pergunta, como por exemplo: "*Quais são os **3** outros processos executados?*" o componente apresenta uma nova resposta e disponibiliza um novo conjunto de questionamentos. A Figura 18 apresenta esta transição para este próximo nível de perguntas.

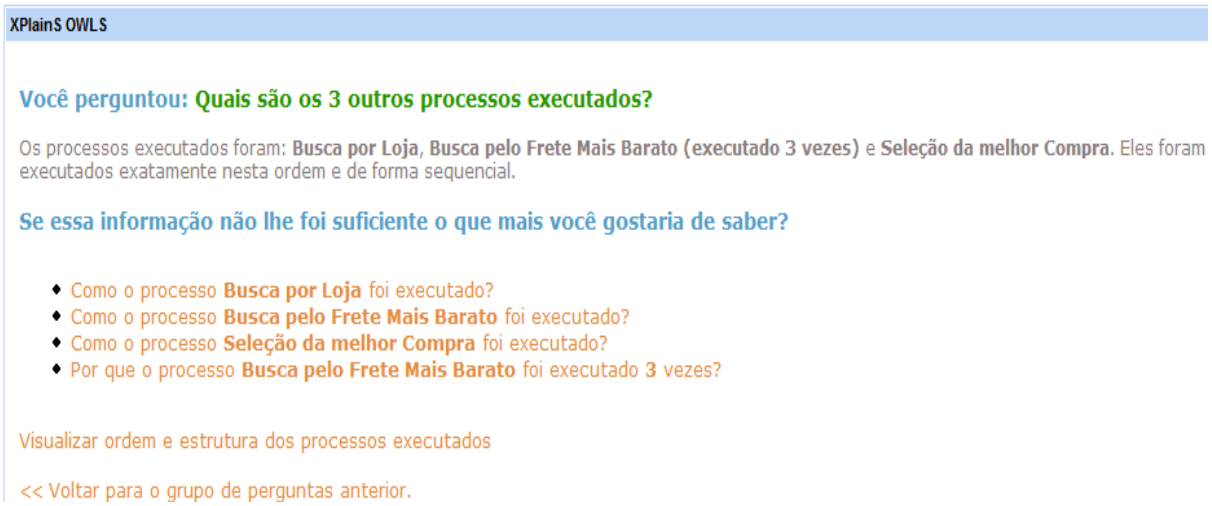

Figura 18 Explicação apresentada depois da seleção da pergunta

4.1.1.2 Componente Histórico de Navegação

O componente histórico de navegação, mostrado em detalhe na Figura 19, apresenta uma estrutura de árvore que contém o histórico de perguntas e respostas já visualizadas pelo usuário. Através deste componente o usuário pode acessar diretamente qualquer uma das perguntas feitas por ele anteriormente sem ter que voltar todos os níveis até o nível da pergunta desejada.

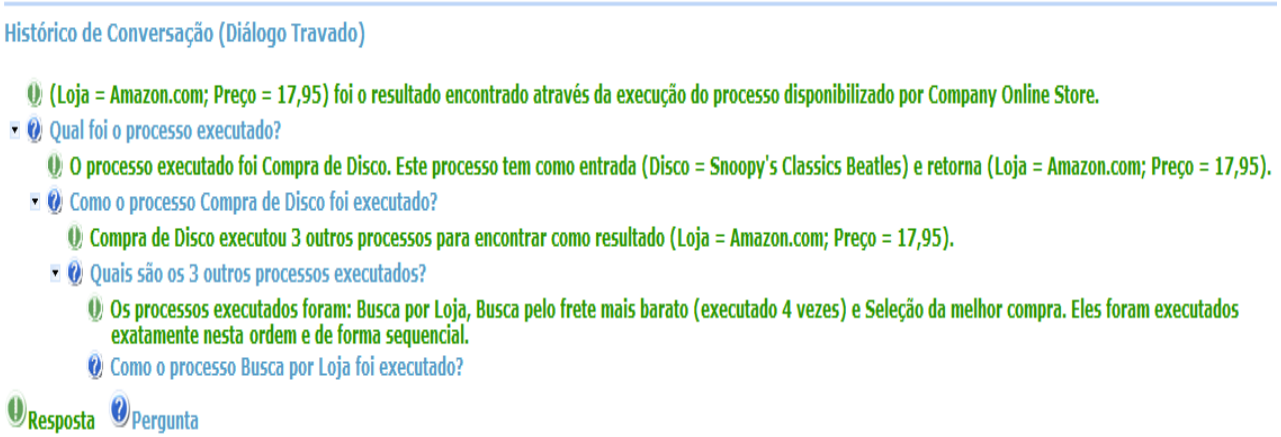

Figura 19 Componente Histórico de Conversação

Ao revisitar uma de suas perguntas, o usuário visualizará novamente o conjunto das possíveis perguntas para aquele nível e poderá escolher qualquer uma delas, criando assim um novo ramo na hierarquia de perguntas do componente. Este novo ramo será adicionado no mesmo nível da pergunta revisitada, conforme mostra a Figura 20, e não ao final da hierarquia.

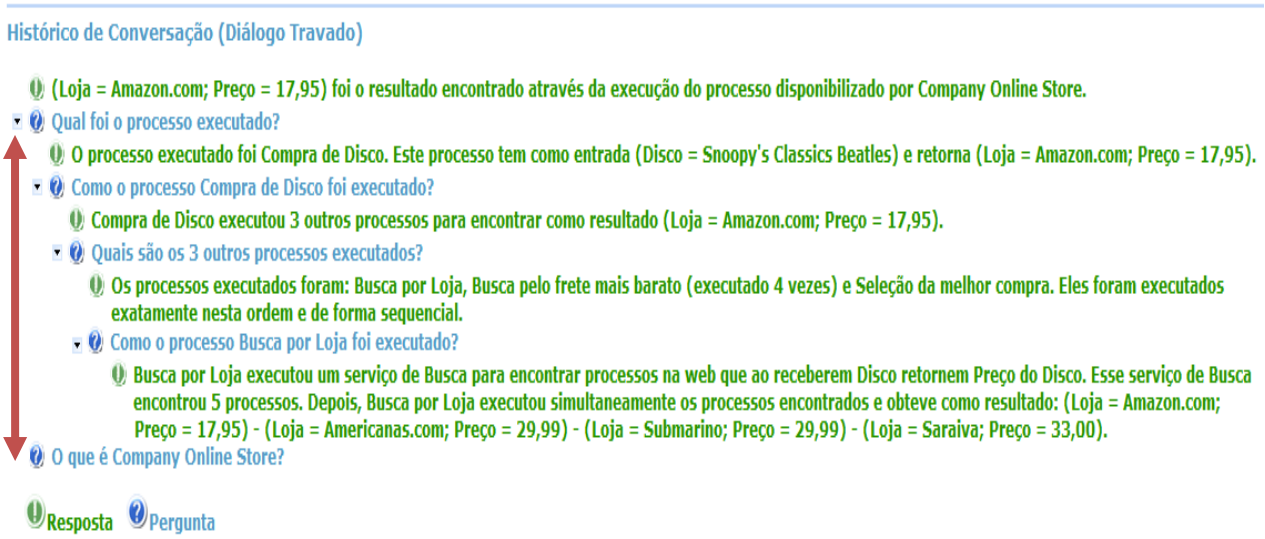

Figura 20 Atualização do histórico após revisita de perguntas

4.1.1.3 Árvore de Processos

O componente de árvore de processos, mostrado em detalhe na Figura 21, exibe a estrutura hierárquica dos processos executados com sucesso ou que apresentaram falhas. Nesta estrutura é exibido apenas o nome dos processos executados. Ao selecionar um processo, o usuário é redirecionado para explicação de como o processo foi executado. Os ícones que representam cada processo indicam se estes foram executados seqüencialmente ou simultaneamente.

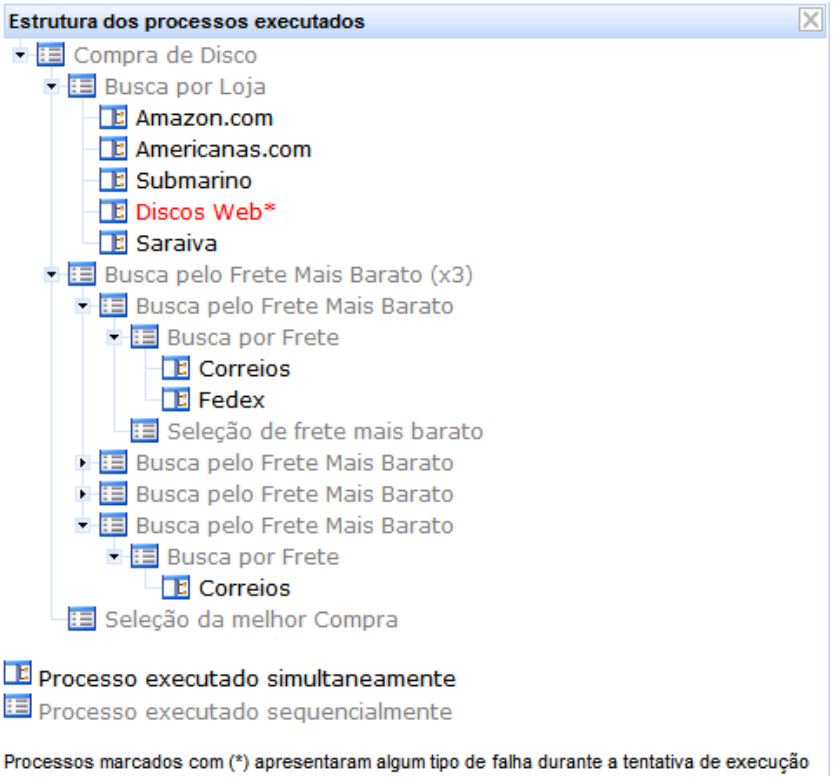

Figura 21 Componente árvore de processos

4.2 Abordagem tecnológica em *XPlainS*

Os três componentes de interface apresentado apresentam as formas que *XPlainS* possui para exibir as informações aos usuários. Para tratar essas informações, *XPlainS* utiliza um conjunto de tecnologias, como: Java, JSP (JBoss RichFaces + JSF), Apache Tomcat e MySQL.

Os critérios para escolha das tecnologias nas quais desenvolvemos *XPlainS* foram robustez, documentação, suporte, portabilidade. Todos os

componentes da arquitetura, com exceção da base de dados, são construídos em Java(Sun Microsystems s.d.).

Java foi escolhida por ser uma linguagem robusta, bem documentada, extensa biblioteca que facilita o uso de recursos de rede, facilidades com internacionalização, código livre, facilidade no desenvolvimento WEB e por ser bem aceita tanto no meio acadêmico quanto no comercial.

As interfaces gráficas foram construídas em JSP (*Java Server Pages)*(Sun Microsystems s.d.)*,* utilizando-se do framework *JBoss RichFaces* (JBoss s.d.) que implementa a especificação JSF (*Java Server Faces*) (Sun Microsystems s.d.) e o Apache Tomcat (The Apache Software Foundation s.d.) que foi escolhido como servidor de aplicação.

Para armazenamento dos dados, MySql (Sun Microsystems s.d.) foi a solução adotada em *XPlainS*, pois é um banco de dados simples, flexível e robusto e como todas as outras ferramentas é gratuito.

5. Avaliação de *XPLAINS*

Neste capítulo, apresentamos as avaliações efetuadas. Primeiro uma avaliação qualitativa através de testes de uso da ferramenta realizados com possíveis usuários de *XPlainS*, a fim de coletar opiniões usadas para verificar o êxito em alcançar os objetivos propostos neste trabalho. Por fim, uma avaliação comparativa com outra ferramenta de explicação ao usuário.

5.1 Avaliação qualitativa

5.1.1 Metodologia

Utilizamos, de forma adaptada, a proposta da pesquisa qualitativa (Denzin e Lincoln 2001) para verificar de forma exploratória a visão dos usuários sobre a proposta da ferramenta. Como fontes de dados, adotamos as técnicas de observação (estruturada) e registro do uso do sistema seguido de entrevista estruturada composta por perguntas mistas (abertas e fechadas). Para o recrutamento dos participantes, adotamos uma amostra pequena (20 participantes) e

proposital, por máxima variação, sendo o fio condutor o perfil de uso de sistemas de internet.

Os perfis dos participantes selecionados foram 10 homens e 10 mulheres com idade variando entre 21 e 53 anos e de idade média de 29,3. Os participantes eram das mais variadas áreas de atuação e com escolaridade entre nível médio e pós-graduação. Este corpus atende ao perfil de usuários que podem a vir fazer uso da ferramenta proposta, pois todos atendiam ao pré-requisito de ter experiência em realizar buscas e compras na internet.

O participante recebeu inicialmente dois documentos. O primeiro era um termo de consentimento, em anexo neste trabalho, onde o participante autorizava sua realização e a permissão para utilização das informações coletadas, seja através da observação da iteração com a ferramenta como também das informações registradas automaticamente durante o uso da ferramenta pelo participante. O segundo documento era um documento impresso que apresentava informações sobre o *XPlainS*, o experimento e o cenário a ser seguido durante o teste. Este documento também se encontra anexado a esta dissertação.

Todas as interações foram assistidas pelo pesquisador e tiveram registradas as informações de quantas vezes o usuário acessava cada um dos componentes disponibilizados por *XPlainS*. Ao término do teste, os participantes foram requisitados a preencher um formulário (também anexado neste trabalho) que contém perguntas que buscavam avaliar o entendimento do objetivo da ferramenta *XPlainS*, pontos positivos e negativos da ferramenta, opiniões e sugestões.

O objetivo desse experimento é tentar validar as seguintes hipóteses sobre a ferramenta desenvolvida:

- Hipótese 1: A ferramenta auxilia o usuário a compreender melhor os resultados obtidos;

- Hipótese 2: A ferramenta aumenta a credibilidade dos resultados encontrados.

5.1.2 Cenário

Todos os participantes receberam um texto que apresentava o seguinte cenário:

> - Você vai comemorar sua união matrimonial e o presente do seu pai será o vinho a ser servido na comemoração. Entretanto, a compra deste vinho deverá ser feita por você. Para isso, você acessa a página http://www.carlosgustavoof.com.br/*XPlainS*Web/simuladorloja que é o endereço de uma loja virtual onde você pode comprar tal produto. Para efetuar a compra, você informa o produto, as características deste produto e os dados para entrega. Esta loja busca na Internet onde o produto encontra-se mais barato e a transportadora que realiza pelo menor preço o transporte do produto. Ao final da compra, você dever justificar a compra como melhor opção para seu pai. Para isso você deverá usar as informações oferecidas pelo sistema *XPlainS*.

Ao término da leitura, o usuário acessaria o endereço do site da loja onde iria realizar a compra. Desenvolvemos uma aplicação que simula esse processo para ser utilizada neste teste. Ao acessar o site da loja, a primeira tela solicita que o usuário informe seu nome para que internamente possamos associar as dados que serão gravados durante sua iteração com *XPlainS*. Ao confirmar o seu nome, o usuário é levado para uma tela onde informa o produto a ser comprado, as características deste produto e os dados para entrega. O usuário confirma e o resultado da consulta lhe é apresentado. A Figura 22 mostra essas duas etapas.

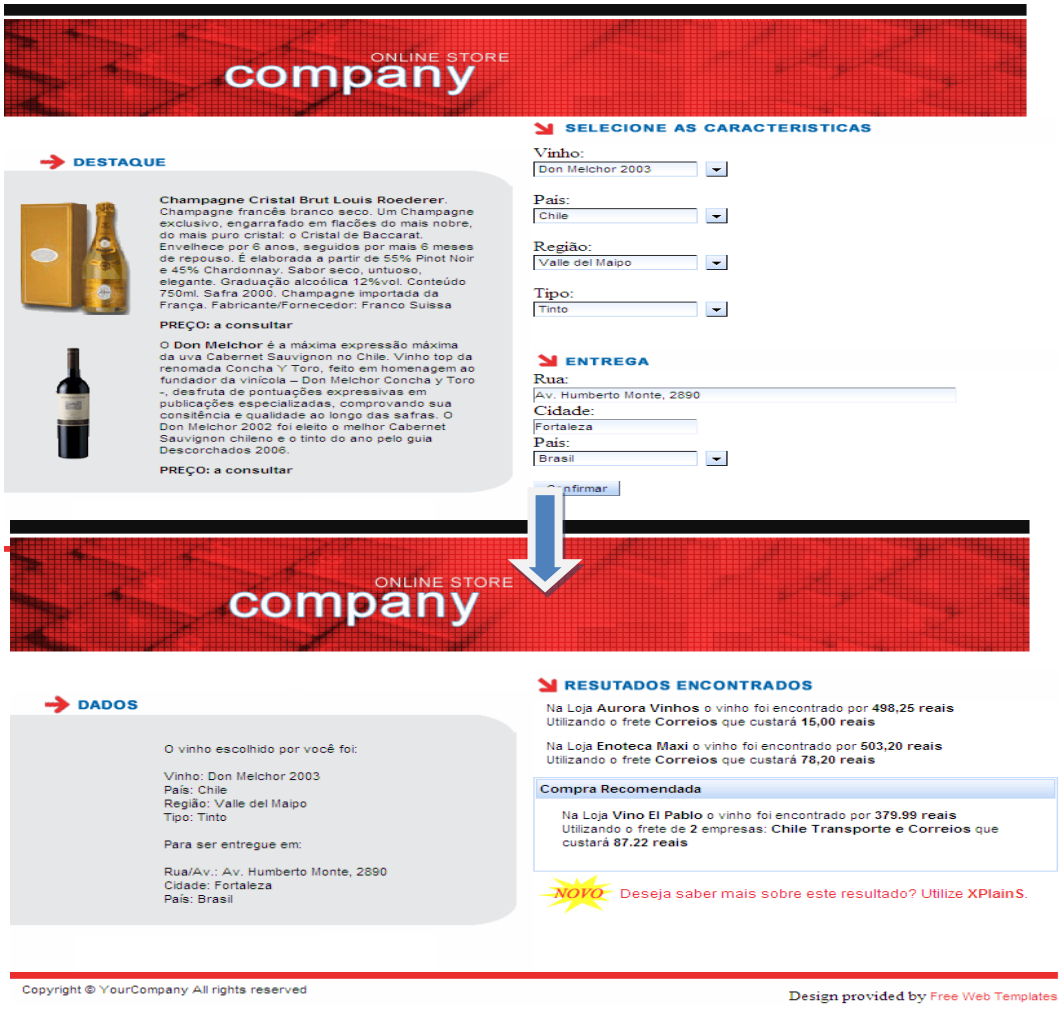

Figura 22 Site utilizado para simular uma loja de busca de produtos

O usuário obtém do site três opções de lojas onde a compra poderia ser realizada e recomenda a última como sendo a melhor opção de compra para o produto pesquisado, levando em consideração o fator menor preço. Para este cenário a opção recomenda é apresentada da seguinte forma:

> *"Na loja **Vino El Pablo** o vinho foi encontrado por **379,99 reais**. Utilizando o frete de 2 empresas: **Chile Transporte** e **Correios** que custará **87,22 reais**."*

Para produzir este resultado, a loja executou três outros serviços. O primeiro serviço era responsável por buscar as lojas que vendem o produto e obteve como resultado 4 lojas. Dentre elas, uma encontrava-se indisponível no momento da

consulta. O próximo serviço tinha como objetivo encontrar, para cada uma das lojas, empresas que realizassem o frete da loja até o endereço do usuário e dentre elas selecionar a que fizesse o frete mais barato. O serviço que busca as empresas para realizar o frete da loja para local de entrega obteve resultado para todas as lojas com exceção da loja Vino El Pablo. Então, este serviço teve que fazer uma composição de duas empresas de frete para que pudesse realizar a entrega desta loja. Por último, foi executado um serviço que somava o preço do produto com o do frete e selecionava o de menor valor. Dessa forma, a Loja Vino El Pablo foi escolhida, pois mesmo com o frete mais caro (realizado por duas empresas) que das outras lojas, o preço final justificou sua escolha.

Esses detalhes do processo não são apresentados ao usuário que obteve apenas o resultado final. Entretanto, na mesma tela em que o resultado é exibido, é também apresentada ao usuário uma opção para que ele possa verificar como o processo de geração do resultado foi obtido. Ao selecionar esta opção, o usuário é redirecionado à ferramenta *XPlainS*.

A partir deste momento, o usuário teve total liberdade para fazer uso da ferramenta, buscando entender como o resultado apresentado no site de compras foi produzido e quais fatores levaram uma das opções a ser recomendada. A Figura 23 mostra três das diversas interações que os usuários poderiam realizar na ferramenta.

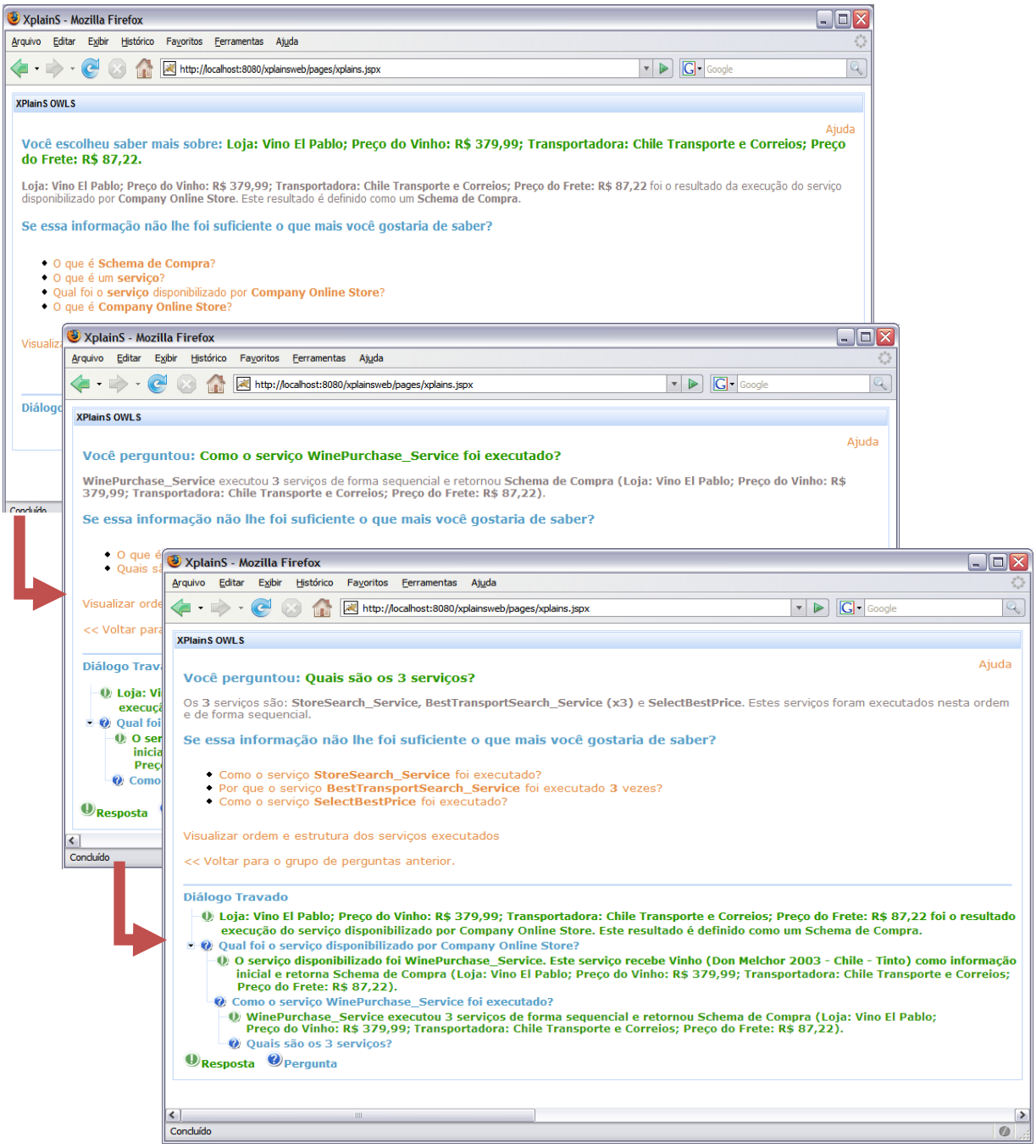

Figura 23 Uso da ferramenta *XPlainS* nos testes

5.1.3 Análise dos Resultados

Após finalizar o período de testes, avaliamos os questionários respondidos pelos participantes e as informações registradas através da interação do usuário com o *XPlainS*. Com isso, buscamos verificar se as hipóteses que defendíamos antes da realização dos testes haviam sido confirmadas ou se foram refutadas.

5.1.3.1 Primeira Hipótese

A primeira hipótese afirma que a ferramenta auxilia o usuário a compreender melhor os resultados obtidos. Para verificarmos a validade da primeira hipótese, buscamos identificar através dos relatórios quatro pontos:

- Entendimento do objetivo da ferramenta;
- Verificar se este objetivo é cumprido;
- Vantagens em utilizar a ferramenta;
- Utilidade das informações apresentadas.

Com a análise dos questionários, identificamos que 70% deles entenderam claramente o objetivo de *XPlainS*, que é mostrar como foi produzido um determinado resultado. Outros 20% entenderam parte deste objetivo e os 10% restante não compreenderam o objetivo de *XPlainS* como esperávamos. Identificamos que metade do grupo que não entendeu a ferramenta fez pouco uso da mesma, isto é, enquanto a média de interação do usuário com o sistema foi de vinte perguntas selecionadas por pessoa, tivemos usuários que declararam que não entenderam a ferramenta e selecionaram somente duas perguntas. A seguir, apresentamos alguns dos relatos que exprimem bem a opinião dos participantes, os três primeiros mostram um total entendimento do objetivo da ferramenta, enquanto o último apresenta um entendimento parcial:

"... ela informa maiores detalhes de como foi feito a busca e não só sobre o produto."

"... ele informa o que está 'por trás' da busca gerada pelo site."

"... busca mostrar como o resultado foi encontrado."

"... explicar melhor a opção de compra."

Dentre os participantes que entenderam o objetivo de *XPlainS*, de forma total ou parcial, 94,44% afirmaram que a ferramenta cumpre totalmente o objetivo a que se propõe e 5,56% dizem que este objetivo é atingindo apenas parcialmente. Na seqüência, apresentamos dois relatos que denotam essas opiniões.

"Sim. Oferece a explicação por trás da utilização dos serviços de busca e sugestões de compras. O objetivo é cumprido de forma satisfatória."

"Mais ou menos. Tenta explicar a estrutura de funcionamento do mecanismo de busca. Cumpre parcialmente, já que explicação não é muito amigável ao usuário."

Em termos gerais esses mesmos participantes citaram como vantagem a disponibilização das informações, pois estas permitiam que o usuário pudesse compreender melhor as etapas de busca e/ou o resultado apresentado, no cenário do teste a melhor loja para realizar compra. Abaixo seguem algumas dessas citações:

"A visualização das etapas de uma busca."

"Fornecer informações sobre a forma de busca."

"Esclarecer dúvidas sobre o resultado apresentado."

"Obter mais detalhes antes da compra."

Sobre a utilidade das informações apresentadas, verificamos que 65% dos participantes acharam úteis as explicações, pois a partir delas puderam entender como o resultado foi produzido. Quando perguntado sobre a utilidade das informações apresentadas pelo *XPlainS*, obtivemos respostas do tipo:

"*Sim, para entender, por exemplo, por que dois fretes foram utilizados.*"

Afirmações como esta foram importantes, pois podemos observar que os usuários conseguiram entender detalhes do cenário, que sem a explicação não seria possível. Outros 20% afirmaram a utilidade da informação, mas não expressaram o motivo. Os 25% restantes afirmaram que as informações não lhe foram úteis, entretanto, com exceção de uma pessoa, essas são as mesmas que não compreenderam o objetivo do sistema.

Com essa análise podemos concluir que o uso da ferramenta *XPlainS* contribuiu para uma melhor compressão do resultado encontrado e isto valida nossa primeira hipótese. Porém, observamos algumas críticas quanto à apresentação das explicações. Mais adiante neste capítulo discutiremos em detalhes este ponto.

5.1.3.2 Segunda Hipótese

Após validar a primeira hipótese, partimos para analisar os fatores relevantes que confirmem a segunda hipótese. Essa hipótese afirma que a ferramenta aumenta a credibilidade dos resultados encontrados. Sobre este aspecto, dois pontos foram analisados:

- A confiança que os participantes têm nos resultados apresentados pelas ferramentas de busca atuais;
- O uso da ferramenta *XPlainS* no seu dia-a-dia.

Em análise, descobrimos que metade dos participantes confia nos resultados que hoje lhe são apresentados pelos sites de busca e a outra metade confia parcialmente. Dos usuários que confiam no que hoje lhe é apresentado, todos, sem exceção, informaram que utilizariam a ferramenta *XPlainS* no seu dia-a-dia, caso ela fosse disponibilizada. Entretanto, um destes participantes que afirmou que utilizaria a ferramenta, foi um dos que não compreendeu corretamente o objetivo de *XPlainS*. Dos participantes que responderam que não confiam totalmente nos resultados apresentados pelas ferramentas de busca disponíveis atualmente, 50% disseram que utilizariam a ferramenta no seu dia-a-dia, 20% disseram que a utilizariam em alguns casos e os 30% restantes disseram que não utilizariam a

ferramenta. Esse resultado, de certa forma, poderia refutar nossa hipótese, pois esperávamos que as pessoas que relataram não acreditar totalmente nos resultados que lhe são apresentados hoje seriam justamente as que fariam uso da ferramenta. Porém, ao analisar em detalhes as respostas dos usuários que afirmaram que não utilizariam a ferramenta, nos deparamos com a seguinte situação: um dos que não utilizaria a ferramenta faz parte do grupo que não compreendeu o objetivo da ferramenta e o restante do grupo remeteu as seguintes opiniões quando indagados sobre se utilizariam a ferramenta:

> *"... o resultado fornecido pelo site de busca (apresentação dos preços do produto encontrado e a indicação da loja que oferece o serviço mais barato) já é suficiente."*
>
> *"Ainda não. Os resultados são interessantes, mas as informações estão estruturadas de forma não muito amigável e as respostas poderiam ser mais claras."*

A partir destes resultados, observamos que todos os participantes que disseram confiar nos resultados que hoje lhe são apresentados na Web e a maior parte daqueles que possuem algum tipo de desconfiança nesses resultados, declararam que fariam uso da ferramenta. Identificamos um grupo que afirmou não confiar totalmente nas ferramentas de busca disponíveis atualmente, mas mesmo assim não utilizaria a nossa ferramenta. Entretanto, esses usuários não relataram que deixariam de utilizar a ferramenta porque ela não agregaria credibilidade às informações, mas sim pelo fato de já ficarem satisfeitos com o que lhe é mostrado pelos sites de busca ou por afirmarem que a ferramenta necessita de mudanças na forma de apresentação das explicações, a fim de se tornar utilizável. Posto isso, podemos concluir que a segunda hipótese também é válida e que novamente críticas à apresentação das informações foram identificadas.

5.1.4 Observações Gerais

Os relatos dos usuários indicam que às vezes a ferramenta apresentava informações não relevantes ou repetia uma mesma pergunta em vários pontos da explicação. Isto, em alguns casos, tornava a explicação tediosa e desviava o foco do usuário na busca da informação desejada. Isto ocorre por que nossa ferramenta é livre de domínio, pois ao passo que não temos ligação nenhuma com o domínio, ficamos incapazes de criar um ranking das perguntas mais relevantes ou até mesmo identificar qual o melhor momento para se mostrar ou não alguma informação. Para conseguirmos identificar quais informações são relevantes ou não para serem apresentadas ao usuário, teríamos que ter conhecimento do domínio ou criarmos um mecanismo para que o próprio usuário pudesse informar qual o tipo de informação que lhe é útil. Outra razão para a ocorrência dessas críticas é que a ferramenta está explicando fluxos que estão sendo descobertos e executados dinamicamente. A explicação desses fluxos é limitada pela incapacidade de se obter informações sobre o domínio previamente.

No teste aplicado, 20% dos participantes relataram críticas referentes à quantidade de informação apresentada pela ferramenta e a irrelevância de algumas informações apresentadas em determinados momentos. Sobre este aspecto surgiram alguns relatos como:

> *"Diminuiria a quantidade de informações sobre o procedimento, pois algumas são informações básicas e pode tornar a pesquisa cansativa"*
>
> *"… Permitir ao usuário escolher o nível de informação que ele deseja, por exemplo: básico, avançado, etc."*
>
> *"… que as definições fiquem separadas para serem utilizadas somente por quem realmente se interessa por essa abordagem"*

Outro problema encontrado diz respeito à dependência da ontologia. Por ser livre de domínio, a única fonte que nossa ferramenta pode utilizar para trazer alguma informação pertencente ao domínio para o usuário é o uso das ontologias de

processos. Tais ontologias descrevem os processos, suas entradas e saída. Com isso, tentamos aumentar o nível de compreensão da explicação apresentada ao usuário. Devido à essa dependência, a qualidade de nossa explicação é diretamente proporcional ao detalhamento da ontologia, ou seja, quanto mais detalhes apresentar a ontologia, mais compreensível se torna a explicação. Alguns usuários relataram dificuldade em utilizar a ferramenta:

> "... como não sou da área de tecnologia, os nomes dos serviços executados pelas empresas estão em outra língua, o que dificultou o meu entendimento."

Esse fato aconteceu porque algumas de nossas ontologias de serviços estavam incompletas e não continham informações, como por exemplo, das descrições dos serviços. Dessa forma, a explicação foi obrigada a utilizar o nome técnico dos serviços no lugar de sua descrição, o que dificultou a compreensão de alguns usuários.

Podemos então concluir que apesar das limitações impostas pelo não conhecimento prévio do domínio e da dependência das ontologias envolvidas no processo, ainda assim o uso da ferramenta de explicação facilita o entendimento do resultado apresentado ao usuário e as informações disponibilizadas dão maior credibilidade ao resultado.

Ao término dessa avaliação, pudemos constatar que alguns dos participantes apresentaram sugestões de melhorias da ferramenta de *XPlainS*, como por exemplo: melhorias de interface, melhorias na apresentação das informações e adição de novas funcionalidades. As melhorias de interface e apresentação das informações já estão concluídas. Contudo, a adição de novas funcionalidades, como, por exemplo, a possibilidade do usuário poder personalizar as informações que ele deseja que sejam apresentadas ou não, ficarão para um trabalho futuro.

5.2 Avaliação comparativa

Para avaliar a qualidade das explicações produzidas por um fluxo de processo que segue nossa abordagem, iremos comparar as explicações geradas por *XPlainS* com a de uma proposta que tem conhecimento prévio do problema.

Para realizar este comparativo, realizamos a modelagem de um processo de configuração de computador baseado em (Friedman 2005) que apresenta o processo de fabricação de computadores Dell. Neste contexto, o cliente faz um pedido de um computador através do telefone ou internet. O pedido do usuário contém as informações necessárias para a montagem, pagamento e entrega. Feito isso, o pedido é analisado pelos profissionais das Dell para que eles possam preparar uma configuração de acordo com as necessidades do cliente e a disponibilidade de peças das seis montadoras espalhadas pelo mundo.

A abordagem de Pinheiro, V C (2005) propõe prover explicações para SBCs desde que utilizem o método de resolução de problemas, como por exemplo, o método *Propose and Revise* (Schreiber, et al. 1994). Por este motivo, modelamos o problema de configuração de computadores utilizando o método *Propose and Revise* onde cada um dos componentes deste método é definido como um serviço Web e estes são compostos para resolver o problema.

A Figura 24 apresenta as explicações geradas por um sistema que utiliza a proposta de geração de explicação de Pinheiro, V C (2005). Nela podemos observar uma explicação estruturada no formato de um problema de configuração.

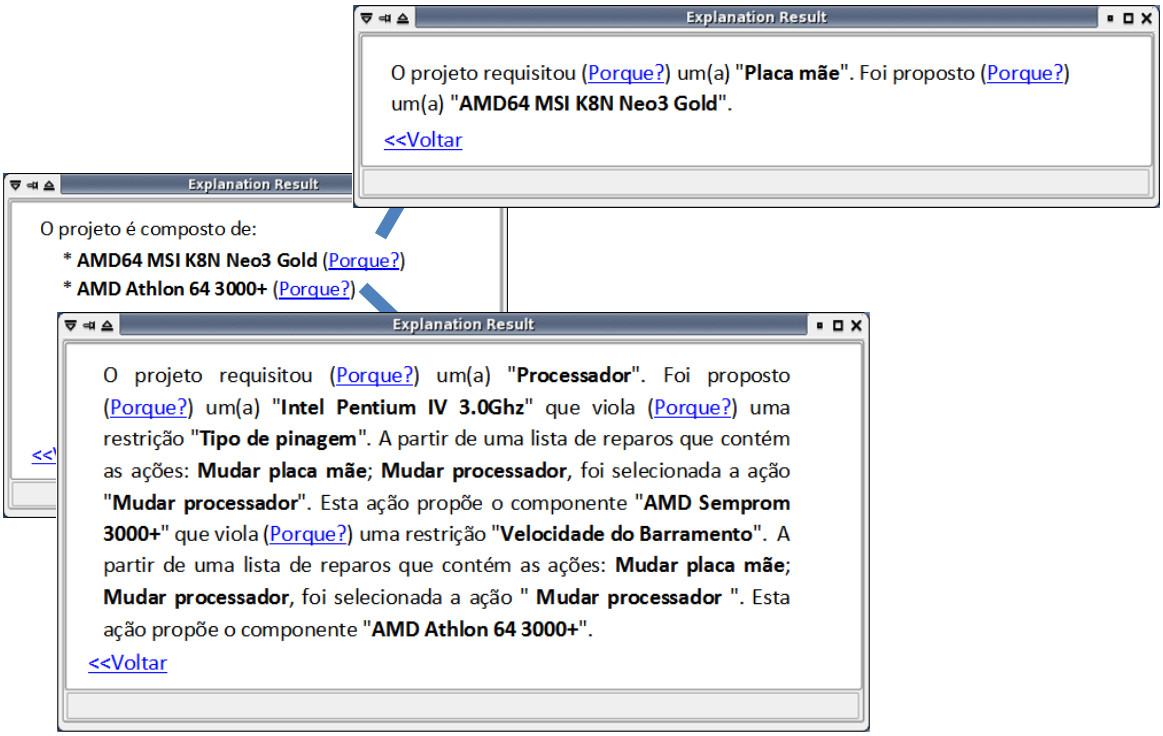

Figura 24 Explicação do processo de configuração de um computador utilizando a proposta de (Pinheiro, V C 2005).

O conhecimento do método de resolução do problema permitiu que a explicação fosse estruturada de forma mais “amigável” e pode utilizar uma linguagem direcionada ao contexto do problema, facilitando a compreensão do usuário.

Diferentemente da metodologia de *XPlainS*, que desconhece tanto o domínio quanto a metodologia utilizada na resolução do problema, as explicações apresentadas por ele apresentam uma linguagem mais estruturada de como o processo foi executado e não do contexto do problema. Entretanto, segundo as avaliações qualitativas realizadas, as explicações geradas por *XPlainS,* apesar de apresentarem menor expressividade do domínio, também permitem que o usuário entenda como o processo ocorreu e como os resultados foram obtidos A Figura 25 apresenta as explicações geradas por *XPlainS*.

Figura 25 Explicação sem Conhecimento do Domínio.

5.2.1 Observações Gerais

Ao observar as explicações geradas por cada uma das propostas, verificamos que o nível de compreensão apresentado nas explicações da primeira proposta é superior ao apresentado por *XPlainS* por dois motivos: pelo uso de uma estruturação direcionada ao método de resolução do problema e pela capacidade de apresentar aspectos do domínio do sistema.

Todavia, as explicações providas por *XPlainS,* por apresentarem uma explicação mais genérica, permitem que esta metodologia de explicação possa ser reutilizada em qualquer domínio. Isso não ocorre com as propostas de explicações que se utilizam do domínio ou forma de resolver o problema para auxiliar na explicação, visto que a cada mudança, as explicações sofrem também mudanças.

6. Conclusão

Neste trabalho, descrevemos um framework que gera um sistema de explicação de fluxos de execução dos serviços Web descritos em uma ontologia de serviços Web. Mostramos a viabilidade de se gerar um conjunto de perguntas e respostas que permitem ao usuário a compreensão das razões pelas quais um resultado em particular foi obtido, por meio da execução de diversos processos distribuídos e compostos dinamicamente. Outro objetivo alcançado foi o desacoplamento das explicações do domínio em que estão sendo aplicadas. Esse desacoplamento permite que alguém que deseje adicionar a característica de explicação a seus processos precise apenas descrever o fluxo de execução de seu processo em uma ontologia de serviços Web e anexar o componente de explicação à API responsável por interpretar esta ontologia de serviços.

Também pudemos concluir, através dos testes realizados com os usuários, que a ferramenta os auxiliou a compreender como o fluxo de serviços foi executado para produzir o resultado informado. Porém, para a grande maioria dos usuários, a utilização de outras características do domínio que pudessem aumentar a complexidade do processo (no caso do treinamento, fatores que influenciassem na escolha do resultado da loja, como por exemplo, tempo de frete, uso de preferências pessoais) tornaria o uso da ferramenta ainda mais interessante. Isso nos leva a concluir que a utilidade da ferramenta é diretamente proporcional ao aumento da complexidade do processo em execução, pois com o aumento da complexidade do processo existe também um aumento na riqueza de informação disponibilizada ao

usuário. Em contrapartida, processos simples podem gerar informações triviais para o usuário.

Ao término deste trabalho, conseguimos apresentar uma solução que contribui tanto em termos acadêmicos quanto em termos técnicos.

Nossa contribuição acadêmica ocorreu pelo fato de que ao realizarmos as pesquisas bibliográficas identificamos limitações nas soluções atuais que se propõem a explicar fluxos de execução de serviços Web semânticos. O uso dessas explicações pode variar de acordo com o perfil de quem as utiliza. Pessoas leigas, usuárias das ferramentas de buscas que são providas pela internet poderão utilizar a ferramenta para o entendimento de como o seu resultado foi encontrado. Por outro lado, a ferramenta também pode auxiliar os usuários que desenvolvem soluções com SWS, pois através da ferramenta eles podem monitorar quais serviços estão sendo executados, identificando serviços que estão falhando ou serviços que deveriam estar produzindo determinado resultado, mas por algum motivo não estão.

Nossa contribuição técnica se resume a forma que desenvolvemos nossa solução. Ao escolhermos propor uma solução de explicação que não tivesse conhecimento do domínio, pudemos desenvolver um *framework* que pode ser reutilizado por aplicações que desejam prover explicações de seus fluxos, sem que estes precisem se modificar para utilizar esta infra-estrutura. Para isso, basta que eles descrevam o fluxo do serviço em uma ontologia de serviços que já tenha a interface de explicação desenvolvida, como, por exemplo, OWL-S que já possui esta interface implementada. Caso seja escolhida uma nova ontologia de representação de SWS, primeiramente, faz-se necessário implementar esta interface na ferramenta que processa a ontologia.

6.1 Trabalhos futuros

Ao término do trabalho, identificamos algumas limitações em nossa solução.

Uma característica importante de uma ferramenta de explicação, que a torna dinâmica e que atrai a utilização do usuário, é a geração de informação sempre relevante ao usuário. *XPlainS* é limitada nesse aspecto, pois nossas explicações por não terem conhecimento do domínio, apresentam algumas vezes informações que talvez não sejam relevantes para alguns usuários. Por isso, acreditamos que uma extensão importante da ferramenta de explicação, para que seja utilizada no dia-a-dia, é a personalização das explicações, ou seja, a customização de quais informações podem ser omitidas ou exibidas de acordo com as características ou preferências do usuário.

Outra extensão para a ferramenta de explicação é a de permitir um nível maior de abstrações das informações apresentadas, pois neste trabalho o único nível de abstração provido é o que trata das repetições de serviços. As repetições são resumidas em informações que dizem que determinado serviço foi executado em um determinado número de vezes. Para isso, vislumbramos uma integração com a ferramenta de suporte a explicações, IW *Abstractor* (Furtado, et al. 2007), provida pelo *framework* IW.

Como citado anteriormente, o uso de fatores que aumentem a complexidade do processo em execução aumenta a riqueza e utilidade da explicação. Por isso, acreditamos que *XPlainS* pode ganhar maior qualidade ao ser estendido para explicar preferências e restrições (Ayres 2007). Atualmente, esses tipos de critérios não são explicáveis por nossa solução.

Por fim, estamos trabalhando para integrar *XPlainS* com o *framework* IW e logo o mesmo estará disponível no site do Inference Web[5].

[5] Inference Web. Para mais informações acesse: http://inference-Web.org/

Bibliografia

Akkiraju, R, et al. “Web Service Semantics - WSDL-S.” *A joint UGA-IBM Technical Note, version 1.0*, 18 de Abril de 2005.

Alesso, H P, e C F Smith. “Developing Semantic Web Services.” *Massachussetts: A K Peters*, 2005.

Antoniou, G, e F Harmelen. *A Semantic Web Primer.* Cambridge, Massachusetts: The MIT Press, 2004.

Ayres, L. “OWLPREF: Uma Representação Declarativa de Preferências Qualitativas para a Web Semântica.” *Dissertação de Mestrado Informática Aplicada - Unifor*, 2007.

Beckett, D, e B McBride. *RDF/XML Syntax Specification (Revised).* 15 de Dezembro de 2003. http://www.w3.org/TR/rdf-syntax-grammar/ (acesso em 10 de Junho de 2008).

Berners-Lee, T, J Hendler, e O Lassila. “The Semantic Web.” *Scientific American*, Maio de 2001.

Booth, D, et al. *Web Services Architecture.* 11 de Fevereiro de 2004. http://www.w3.org/TR/ws-arch/ (acesso em 3 de Junho de 2008).

Bray, T, J Paoli, Sperberg-McQueen C M, E Maler, F Yergeau, e J Cowan. *Extensible Markup Language (XML) 1.1 (Second Edition).* 16 de Agosto de 2006. http://www.w3.org/TR/xml11/ (acesso em 10 de Junho de 2008).

Brickley, D, R V Guha, e B McBride. *RDF Vocabulary Description Language 1.0: RDF Schema.* 10 de Fevereiro de 2004. http://www.w3.org/TR/rdf-schema/ (acesso em 10 de Junho de 2008).

Buchanan, B G, e E H Shortliffe. “Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.” *Addison-Wesley*, 1984.

Christensen, E, F Curbera, G Meredith, e S Weerawarana. *Web Services Description Language (WSDL) 1.1.* 15 de Março de 2001. http://www.w3.org/TR/wsdl (acesso em 7 de Junho de 2008).

—. *Web Services Description Language (WSDL) 1.1.* 11 de Março de 2001. http://www.w3.org/TR/wsdl (acesso em 19 de Maio de 2008).

Clancey, W J. “From GUIDON to NEOMYCIN and HERACLES in twenty short lessons.” *AI Magazine*, 1986: 40-60.

Denzin, N K, e Y S Lincoln. *Handbook of qualitative research.* 2ª Edição. CA: Sage, 2001.

Farrell, J, e H Lausen. *Semantic annotations for wsdl and xml schema.* 2006. http://www.w3.org/2002/ws/sawsdl/spec/ (acesso em 02 de Junho de 2008).

Ferreira, Aurelio B. Holanda. *Mini Aurélio o Dicionário da Língua Portuguesa.* Positivo Editora, 2006.

Fikes, R, P Hayes, e I Horrocks. “OWL-QL - A Language for Deductive Query Answering on the Semantic Web.” *KSL Technical Report 03-14*, 2003.

Freitas, F. L. “Ontologias e a Web Semântica.” *Anais do XXIII Congresso da Sociedade Brasileira de Computação (Vols. 8 - Jornada de Mini-Cursos em Inteligência Artificial)*, 2003: 1-52.

Friedman, T. “The World is flat - A brief history of Twenty-First Century.” *Farrar, Straus and Giroux*, 2005.

Furtado, V, et al. “Abstracting Web Agent Proofs into Human-Level Justifications.” *Proceedings of the 20th FLAIRS Conference, AAAI*, 2007.

Glass, A, e D McGuinness. “Introspective Predicates for Explaining Task Execution in CALO.” *Technical Report KSL-06-04. Knowledge Systems. Artificial Intelligence Laboratory*, 2006.

Gruber, T R. “Toward principles for the design of ontologies used for knowledge sharing.” *International Journal of Human-Computer Studies, 43 (4-5)*, 1995: 907-928.

Gruber, Thomas R. “Encyclopedia of Database Systems.” In: *Springer-Verlag*, por Ling Liu and M. Tamer Özsu (Eds.). 2008.

Gudgin, M, et al. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).* 27 de Abril de 2007. http://www.w3.org/TR/soap12-part1/ (acesso em 15 de Junho de 2008).

Heflin, J. *OWL Web Ontology Language.* 10 de Fevereiro de 2004. http://www.w3.org/TR/2004/REC-webont-req-20040210/ (acesso em 13 de Junho de 2008).

Hendler, J, Berners-Lee T, e E Miller. “Integrating Applications on the Semantic Web.” *Journal of the Institute of Electrical Engineers of Japan, Tóquio*, Outubro de 2002: 676-680.

IBM. *BPWS4J: A platform for creating and executing BPEL4WS processes.* 9 de Agosto de 2002. http://www.alphaworks.ibm.com/tech/bpws4j (acesso em 10 de Junho de 2008).

JBoss. *JBoss RichFaces.* http://www.jboss.org/jbossrichfaces/.

Lieberman, H, e E Wagner. “End-User Debugging for E-Commerce.” *ACM Conference on Electronic Commerce*, Junho de 2003.

M. Paolucci, T, T Kawamura, R Payne, e K. Sycara. “Semantic Matching of Web Services Capabilities.” *In Proceedings of the First International Semantic Web Conference*, 2002: 333-347.

Mandell, D J, e S A McIlraith. “ Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation.” *Proceedings of the Second International Semantic Web Conference (ISWC2003)*, 2003.

Martin, D, et al. “Bringing Semantics to Web Services: The OWL-S Approach.” In: *Semantic Web Services and Web Process Composition*, por Lecture Notes in Computer Science, 26-42. Springer Berlin / Heidelberg, 2005.

—. *OWL-S: Semantic Markup for Web Services.* 22 de Novembro de 2004. http://www.w3.org/Submission/OWL-S/ (acesso em 1 de Junho de 2008).

McGuinness, D. “Explaining Subsumption in Description Logics.” *Rutgers University Thesis*, 1996.

McGuinness, D, A Glass, M Wolverton, e P Pinheiro da Silva. “A Categorization of Explanation Questions for Task Processing Systems.” *Proceedings of the 2007 Workshop on Explanation-aware Computing (ExaCt-2007)*, 2007.

McGuinness, D, D Mandell, S McIlraith, e P Pinheiro da Silva. “Explainable Semantic Discovery Services.” *Stanford Networking Research Center Project Review*, Fevereiro de 2005.

McGuinness, D, E Hsu, J Jenkins, R McCool, S McIlraith, e P Pinheiro da Silva. *KSL Wine Agent.* 2003. http://ksl.stanford.edu/people/dlm/webont/wineAgent/ (acesso em 13 de Junho de 2008).

McGuinness, D, e P Pinheiro da Silva. “Explaining Answers from the Semantic Web.” *The Inference Web Approach. Journal of Web Semantics*, Outubro de 2004: 397-413.

McGuinness, D. L, e F van Harmelen. *OWL Web Ontology Language.* 10 de Fevereiro de 2004. http://www.w3.org/2003/Talks/0818-msm-ws/ (acesso em 7 de Junho de 2008).

McIlraith, S, D Mandell, e H Mi. *Semantic Discovery Service (SDS) Project.* 2003. http://www.ksl.stanford.edu/sds/ (acesso em 10 de Junho de 2008).

McIlraith, S, T C Sont, e H Zeng. “Semantic Web Services.” *IEEE Intelligent Systems,*, Maio/Abril de 2001: 46-53.

McIlraith, S A, T C Son, e H Zeng. “Semantic Web Services.” *IEEE Intelligent Systems. Special Issue on the Semantic Web*, 2001: 46-53.

Mindswap. 2004. http://www.mindswap.org/2004/owl-s/api/ (acesso em 1 de Junho de 2008).

Morley, D., Myers, K. “The SPARK agent framework.” *AAMAS-04 Proceedings*, 2004: 714-721.

Motik, B. “On the Properties of Metamodeling in OWL.” *(Springer, Ed.) Proceedings of the 4th International Semantic Web Conference , 3729*, Novembro de 2005: 548-562.

OASIS. *OASIS Web Services Business Process Execution Language (WSBPEL) TC.* 30 de Julho de 2000. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel (acesso em 10 de Junho de 2008).

PAGELS, M. *The DARPA Agent Markup Language Homepage.* Agosto de 2000. http://www.daml.org/ (acesso em 1 de Junho de 2008).

Perojo, K R, e R R León. “Web Semántica: un nuevo enfoque para la organización y recuperación de información en la web.” *Acimed*, Novembro-Dezembro de 2005.

Pinheiro, P, D L McGuiness, e R Fikes. “A Proof Markup Language for Semantic.” *Technical Report KSL-04-01, Knowledge Systems Laboratory, Stanford University, USA*, 2004.

Pinheiro, V C. “Construção de sistemas baseados em conhecimento interativos: arquitetura e processo de desenvolvimento usando padrões de interação.” *Dissertação de Mestrado Informática Aplicada - Unifor*, 2005.

Rajasekaran, P, J Miller, K Verma, e A Sheth. “Enhancing Web Services Description and Discovery to Facilitate Composition.” *International Workshop on Semantic Web Services and Web Process Composition*, 2004.

Richard, F, J Jenkins, e G Frank. “JTP: A System Architecture and Component Library for Hybrid Reasoning.” *Proceedings of the Seventh World Multiconference on Systemics, Cybernetics, and Informatics*, 27-30 de Junho de 2003.

Roman, D, et al. “Web Service Modeling Ontology.” In: *Applied Ontology, 1(1)*, 77 - 106. 2005.

ROY, J, e A RAMANUJAN. “Understanding web services.” *IEEE Internet Computing*, Novembro - Dezembro de 2001: 69 – 73.

Schreiber, G., B. Wielinga, R. de Hoog, H. Akkermans, e W. Van de Velde. “CommonKADS: A Comprehensive Methodology for KBS Development.” 28-37. 1994.

Shankar, R D, S W TU, e M A Musen. “A Declarative Explanation Framework That Uses a Collection Of Visualization Agents.” *Stanford Medical Institute.*, 1998.

Sirin, E, B Parsia, e J Hendler. “Template-based Composition of Semantic Web Services.” *AAAI Fall Symposium on Agents and the Semantic Web*, Novembro de 2005.

Sirin, E, J Hendler, e B Parsia. “Semi-Automatic Composition of Web Services using Semantic Descriptions.” *Proceedings of the Workshop on Web Services: Modeling, Architecture and Infrastructure*, 2003.

Sperberg-McQueen, C. M. *Web Services and the W3C.* 18 de Agosto de 2003. http://www.w3.org/2003/Talks/0818-msm-ws/ (acesso em 7 de Junho de 2008).

Sun Microsystems. *Java Homepage.* http://www.java.com.

—. *JavaServer Faces Technology.* http://java.sun.com/javaee/javaserverfaces/.

—. *JavaServer Pages Technology.* http://java.sun.com/products/jsp/.

—. *MySQL: The world's most popular open source database.* http://www.mysql.com/.

The Apache Software Foundation. *Apache Tomcat.* http://tomcat.apache.org/.

Vaculin, R, e K Sycara. “Semantic Web Services Monitoring: An OWL-S based Approach.” *Hawaii International Conference on System Sciences*, Janeiro de 2008.

Voormann, H. *Wikipedia: Web Service.* Outubro de 2006. http://en.wikipedia.org/wiki/Web_services (acesso em 2 de Junho de 2008).

W3C. *W3C Semantic Web Activity.* Março de 2007. http://www.w3.org/2001/sw/ (acesso em 20 de Junho de 2008).

Apêndice

Apêndice A – Documentos utilizados na aplicação dos testes

Célula de Pesquisa em Engenharia do Conhecimento

Centro de Ciências Tecnológicas, UNIFOR
Rua Washington Soares, 1321
Edson Queiroz – Fortaleza – CE – 60811-90
Tel. (85) 3477-3287

Termo de Consentimento

Termo de consentimento para realização de experimentos com pessoas na área de informática.

O objetivo desse experimento é observar questões de usabilidade, e aplicabilidade da proposta apresentada no trabalho de mestrado intitulado: GERAÇÃO DE EXPLICAÇÕES PARA SERVIÇOS WEB SEMÂNTICOS EM OWL-S, proposto pelo aluno do mestrado em Informática Aplicada da UNIFOR, Carlos Gustavo Oliveira Fernandes.

Por estas razões, solicitamos seu consentimento para observarmos sua interação (uso) com o protótipo do sistema durante a realização de algumas atividades pré-programadas por este experimento, para a realização de entrevista, bem como a gravação da interação e da entrevista. Para tanto, é importante que você tenha algumas informações adicionais:

a) Os dados coletados durante a observação e a entrevista destinam-se **estritamente** a atividades de pesquisa e desenvolvimento.

b) A divulgação destes resultados pauta-se no **respeito à sua privacidade** e o **anonimato** dos mesmos é preservado em quaisquer documentos que elaboramos.

c) O consentimento para as atividades supracitadas é uma escolha livre, feita mediante a prestação de todos os esclarecimentos necessários sobre a pesquisa.

d) Você tem toda liberdade para interromper esta observação e/ou a entrevista no momento em que desejar.

e) Nossa equipe encontra-se disponível para contato através do telefone 3477-3287, ou pelo email pessoal do pesquisador responsável Carlos Gustavo Oliveira Fernandes.

De posse das informações acima, gostaríamos que você se pronunciasse acerca da observação e da entrevista.

( ) Dou meu consentimento para sua realização.

( ) Não autorizo sua realização.

Fortaleza, _____

Participante	Pesquisador
Nome: _____	Nome: _____
Assinatura:_____	Assinatura:_____

Sobre *XPlainS*:

XPlainS é um sistema que provê explicações da execução de serviços na Web. ***XPlainS*** tem como estratégia gerar informações acerca da execução dos serviços executados para atender sua consulta. A ferramenta fornece ao usuário diálogos de explicação que buscam facilitar a compreensão do resultado apresentado pelo sistema.

Este experimento:

Neste experimento você vai fazer uso da ferramenta ***XPlainS*** com base no cenário descrito a seguir. Após o experimento prático será apresentado um questionário que nos auxiliará a avaliar a proposta.

Cenário:

Você vai comemorar sua união matrimonial e o presente do seu pai será o vinho a ser servido na comemoração. Entretanto, a compra deste vinho deverá ser feita por você. Para isso, você acessa a página http://www.carlosgustavoof.com.br/*XPlainS*Web/simuladorloja que é o endereço de uma loja virtual onde você pode comprar tal produto. Para efetuar a compra, você informa o produto, as características deste produto e os dados para entrega. Esta loja busca na Internet onde o produto encontra-se mais barato e a transportadora que realiza pelo menor preço o transporte do produto.

Ao final da compra, você deve justificar a compra como melhor opção para seu pai. Para isso você deverá usar as informações oferecidas pelo sistema ***XPlainS***.

O site foi indicado por um amigo porque ele disponibiliza a ferramenta ***XPlainS*** para facilitar a compreensão dos resultados de sua consulta, porém você não só não conhece a ferramenta como também não sabe sobre sua eficiência.

Para efetuar a compra, siga TODAS as instruções abaixo.

Instruções para a compra do produto:

1. Informar que deseja comprar um vinho e confirmar.
2. Informar as características do o vinho que deseja comprar:
    a. Nome do Vinho: Don Melchor 2003;
    b. País: Chile;
    c. Tipo: Tinto;
    d. Região: Valle Del Maipo.
3. Em seguida preencher os dados para entrega:
    a. Rua/Avenida: Ao seu critério
    b. Cidade: Fortaleza;
    c. País: Brasil.
4. Confirmar as informações e esperar o resultado. Para qualquer outra opção, não existirá produto disponível.
5. O resultado é apresentado na tela.
6. Como você justificaria o resultado para o seu pai?

Questionário

Dados do participante:

Idade:

Sexo:

Escolaridade:

Profissão:

Sobre sua experiência com a internet:

1) Você usa internet (marque apenas uma das opções):

☐ Mensalmente ☐ Semanalmente ☐ Diariamente ☐ Várias vezes ao dia

2) Você usa sites de busca/pesquisa? Com que freqüência?

3) Você confia nos resultados apresentados por estes sites de busca/pesquisa e se utiliza destes resultados para suas compras?

Sobre o *XPlainS*

4) Você utilizaria o sistema *XPlainS* no seu dia-a-dia caso ele fosse oferecido em seu site de pesquisa/compra? Por quê?

5) Você percebeu diferenças entre *XPlainS* em relação a outras ferramentas de busca/compra? Quais?

6) Qual seria a principal vantagem em se utilizar o *XPlainS*?

7) Ficou claro para você qual o objetivo de *XPlainS*? Qual seria este objetivo para você? O sistema cumpre este objetivo a que se propõe?

Sobre a interação (teste)

8) Você entendeu porque e como o resultado apresentado foi obtido?

9) O sistema oferece informações para que você justifique a compra do produto?

10) Você ficou satisfeito com o resultado? Por quê?

11) O resultado apresentado é realmente o melhor dentre os pesquisados? Você saberia dizer o porquê com base no que foi apresentado pelo *XPlainS*?

12) Qual a melhor opção dentre as não selecionadas pelo Sistema? Por quê?

13) Você teve alguma dificuldade em usar o sistema? Qual?

14) As informações (explicações) apresentadas pelo *XPlainS* lhe foram úteis? Por quê?

15) O que você mudaria (incluiria, tiraria, melhoraria) no sistema *XPlainS*? Por quê?

Comentários:

APÊNDICE B – Regras

&carlosgustavo; = http://carlosgustavoof.com.br/registry/DPR/

Regra	OrderedBefore
Nome:	Task Ordered Before
Descrição:	Representa que um processo P-1 antecede um processo P
Exemplo:	OrderedBefore(P-1, P)
URI:	&carlosgustavo;OrderedBefore.owl#OrderedBefore
Regra	Successful
Nome:	Task Finished Successfully
Descrição:	Representa que uma tarefa P foi finalizada com sucesso
Exemplo:	Successful (P)
URI:	&carlosgustavo;Successful.owl#Successful
Regra	CompletedPredecessors
Nome:	Task Completed Predecessors
Descrição:	Representa que todos os processos antecedentes a P foram completados com sucesso
Exemplo:	∀y OrderedBefore(P-1, P) and Successful (P-1) => CompletedPredecessor(P)
URI:	&carlosgustavo;BottomLevelTask.owl#BottomLevelTask
Regra	ParentOf
Nome:	Task Parent Of
Descrição:	Representa que a tarefa P é super-tarefa (pai) de P-1
Exemplo:	ParentOf(P-1, P)
URI:	&carlosgustavo;ParentOf.owl#ParentOf
Regra	TopLevelGoal
Nome:	Task Top Level Goal
Descrição:	Representa o objetivo de mais alto nível do processo P
Exemplo:	TopLevelGoal(P)
URI:	&carlosgustavo;TopLevelGoal.owl#TopLevelGoal
Regra	Supports
Nome:	Task Parent Support
Descrição:	Representa que o processo P suporta o processo P-1
Exemplo:	Supports(P, P-1)
URI:	&carlosgustavo;ParentSupport.owl#ParentSupport
Regra	SupportsTopLevelGoal
Nome:	Task Top-Level Goal Support
Descrição:	Representa que um processo P suporta o objetivo de mais alto nível
Exemplo:	TopLevelGoal(P) and Supports(P, P-1) => SupportsTopLevelGoal(P)
URI:	&carlosgustavo;TopLevelGoalSupport.owl#TopLevelGoalSupport

Regra	Supports
Nome:	Task Self Support
Descrição:	Representa que o processo P suporta a si mesmo
Exemplo:	Supports(P, P)
URI:	&carlosgustavo;SelfSupport.owl#SelfSupport
Regra	ImplementingProcedureInstance
Nome:	Task Implementing Procedure Instance
Descrição:	Representa que P implementa o procedimento Prc
Exemplo:	ImplementingProcedureInstance(P, Prc)
URI:	&carlosgustavo;ImplementingProcedureInstance.owl#ImplementingProcedureInstance
Regra	ProcedurePreconditionsMet
Nome:	Task Meet Procedure Preconditions
Descrição:	Representa que as precondições do procedimento Prc foram encontradas
Exemplo:	ProcedurePreconditionsMet(Prc)
URI:	&carlosgustavo;ProcedurePreconditionsMet.owl#ProcedurePreconditionsMet
Regra	IntentionPreconditionsMet
Nome:	Task Meet Intention Preconditions
Descrição:	Representa que um processo P teve todas suas precondições atendidas.
Exemplo:	IntentionPreconditionsMet(P)
URI:	&carlosgustavo;IntentionPreconditionMet.owl#IntentionPreconditionMet
Regra	TerminationConditionsNotMet
Nome:	Task Not Meet Termination Conditions
Descrição:	Representa que as condições de término de um processo P não foram atendidas
Exemplo:	TerminationConditionsNotMet(P)
URI:	&carlosgustavo;TerminationConditionsNotMet.owl#TerminationConditionsNotMet
Regra	InstancePrecondition
Nome:	Task Instance Precondition
Descrição:	Representa que PreCond é uma precondição do processo P
Exemplo:	InstancePrecondition(P, PreCond)
URI:	&carlosgustavo;InstancePrecondition.owl#InstancePrecondition
Regra	Met
Nome:	Condition Met
Descrição:	Representa que Cond foi uma condição encontrada

Exemplo:	Met(Cond)
URI:	&carlosgustavo;Met.owl#Met
Regra	BottomLevelTask
Nome:	Task Bottom Level
Descrição:	Repredenta que P é uma tarefa de mais baixo nível
Exemplo:	BottomLevelTask(P)
URI:	&carlosgustavo;BottomLevelTask.owl#BottonLevelTask
Regra	Unfinished
Nome:	Task Unfinished
Descrição:	Representa que o processo P não foi finalizado
Exemplo:	Unfinished(P)
URI:	&carlosgustavo;Unfinished.owl#Unfinished
Regra	Failed
Nome:	Task failed
Descrição:	Representa que o processo P falhou
Exemplo:	Failed(P)
URI:	&carlosgustavo;Failed.owl#Failed
Regra	Future
Nome:	Task execute in future
Descrição:	Representa que o processo P ainda vai ser executado
Exemplo:	Future(P)
URI:	&carlosgustavo;Future.owl#Future
Regra	InstanceTerminationCondition
Nome:	Instance Termination Condition
Descrição:	Representa que Cond é uma condição de término do processo P
Exemplo:	InstanceTerminationCondition(P, Cond)
URI:	&carlosgustavo;InstanceTerminationCondition.owl# InstanceTerminationCondition
Regra	TerminationConditionsMet
Nome:	Task Meet Termination Conditions
Descrição:	Representa que as condições de término de um processo P foram encontradas
Exemplo:	TerminationConditionMet(P)
URI:	&carlosgustavo;TerminationConditionMet.owl#TerminationConditionsMet
Regra	Finished
Nome:	Task finished
Descrição:	Representa que o processo P foi finalizado
Exemplo:	Finished(P)
URI:	&carlosgustavo;Finished.owl#Finished

Regra	Atomic
Nome:	Atomic Procedure
Descrição:	Representa que o procedimento Proc é de execução atômica
Exemplo:	Atomic(Proc)
URI:	&carlosgustavo;Atomic.owl#Atomic
Regra	Sequence
Nome:	Sequence Procedure
Descrição:	Representa que o procedimento Proc é executado como uma seqüência
Exemplo:	Sequence(Proc)
URI:	&carlosgustavo;Sequence.owl#Sequence
Regra	Repeat
Nome:	Repeat Procedure
Descrição:	Representa que o procedimento Proc é executado como uma repetição
Exemplo:	Repeat(Proc)
URI:	&carlosgustavo;Repeat.owl#Repeat
Regra	Parallel
Nome:	Parallel Procedure
Descrição:	Representa que o procedimento Proc é executado em paralelo e que não precisa esperar todos finalizarem a execução para continuar
Exemplo:	Parallel(Proc)
URI:	&carlosgustavo;Parallel.owl#Parallel
Regra	ParallelJoin
Nome:	Parallel Join Procedure
Descrição:	Representa que o procedimento Proc é executado em paralelo e que precisa esperar todos finalizarem a execução para continuar
Exemplo:	ParallelJoin(Proc)
URI:	&carlosgustavo;ParallelJoin.owl#ParallelJoin
Regra	SequenceWithoutOrder
Nome:	Sequence Without Order Procedure
Descrição:	Representa que o procedimento Proc é executado como uma seqüência, mas aleatoriamente
Exemplo:	SequenceWithoutOrder(Proc)
URI:	&carlosgustavo;SequenceWithouOrder.owl#SequenceWithouOrder
Regra	Choice
Nome:	Choice Procedure
Descrição:	Representa que o procedimento Proc foi escolhido para ser

	executado dentro uma lista de outros
Exemplo:	Choice(Proc)
URI:	&carlosgustavo;Choice.owl#Choice
Regra	InsideRepeat
Nome:	Inside Repeat Task
Descrição:	Representa que se um processo antecedente ao processo P, que seja uma repetição, não tenha sido finalizado então P faz parte dessa repetição
Exemplo:	∀Pr OrderedBefore(Pr, x) and Repeat(Pr) and Unfinished(Pr) => InsideRepeat(P)
URI:	&carlosgustavo;InsideRepeat.owl#InsideRepeat
Regra	InsideParallel
Nome:	Inside Parallel Task
Descrição:	Representa que se um processo antecedente ao processo P for um processo paralelo e não tenha sido finalizado então P faz parte desse processo paralelo
Exemplo:	∀Pr OrderedBefore(Pr, P) and (Parallel(Pr) or ParallelJoin(Pr)) and Unfinished(Pr) => InsideParallel(P)
URI:	&carlosgustavo;InsideRepeatJoin.owl#InsideRepeatJoin
Regra	InsideSequence
Nome:	Inside Sequence Task
Descrição:	Representa que se um processo antecedente ao processo P for uma seqüência e não tenha sido finalizado então P faz parte dessa seqüência
Exemplo:	∀Pr OrderedBefore(Pr, P) and (Sequence(Pr) or SequenceWithouOrder(Pr)) and Unfinished(Pr) => InsideSequence (P)
URI:	&carlosgustavo;InsideSequence.owl#InsideSequence

APÊNDICE C – Javadoc das principais classes implementadas e modificadas em *XPlainS*

br.mia.unifor.sws

Interface ExplanationInterface

All Superinterfaces:
org.mindswap.owls.process.execution.ProcessMonitor
All Known Implementing Classes:
ExplanationOWLS

public interface **ExplanationInterface**
extends org.mindswap.owls.process.execution.ProcessMonitor
Interface para monitorar as funçoes de OWL-S. Realiza as gravações de *XPlainS* Foi estendido a interface de monitoramente de execucao de OWL-S.

Method Summary

void	**addCondition**(org.mindswap.owls.process.Condition condition) Método responsável por capturar a condicao e armazenar na pilha
void	**endProcess**(org.mindswap.owls.process.Process process) Metodo responsavel por capturar o processo que finalizou a execucao
void	**failedProcess**(org.mindswap.owls.process.Process process) Metodo responsavel por capturar o processo que falhou durante a execucao
void	**invokeWebService**(javax.xml.soap.SOAPMessage soap) Metodo responsável por anexar o ultimo no de prova na invocacao do WS
void	**startProcess**(org.mindswap.owls.process.Process process, org.mindswap.query.ValueMap inputs) Metodo responsavel por capturar o processo que iniciou a execucao

Methods inherited from interface org.mindswap.owls.process.execution.ProcessMonitor
executionFailed, executionFinished, executionFinished, executionStarted, executionStarted, getMonitorFilter, setMonitorFilter

Method Detail

addCondition

void **addCondition**(org.mindswap.owls.process.Condition condition)

Método responsável por capturar a condicao e armazenar na pilha

Parameters:

condition -

---

endProcess

void **endProcess**(org.mindswap.owls.process.Process process)

Metodo responsavel por capturar o processo que finalizou a execucao

Parameters:

process - Processo que esta finalizado a execucao

---

failedProcess

void **failedProcess**(org.mindswap.owls.process.Process process)

Metodo responsavel por capturar o processo que falhou durante a execucao

Parameters:

process - Processo que ocoreu falhou durante a execucao

---

invokeWebService

void **invokeWebService**(javax.xml.soap.SOAPMessage soap)

Metodo responsável por anexar o ultimo no de prova na invocacao do WS

Parameters:

soap - Mensagem utilizada para anexar o ultimo no da prova

---

startProcess

void **startProcess**(org.mindswap.owls.process.Process process,
org.mindswap.query.ValueMap inputs)

Metodo responsavel por capturar o processo que iniciou a execucao

Parameters:

process - Processo que iniciou a execucao

inputs - Parametros de entrada para que o processo seja executado

---

br.mia.unifor.owls

Class ExplanationOWLS

java.lang.Object
  └ **br.mia.unifor.owls.ExplanationOWLS**

All Implemented Interfaces:

ExplanationInterface, org.mindswap.owls.process.execution.ProcessMonitor

public class **ExplanationOWLS**
extends java.lang.Object
implements ExplanationInterface

Field Summary

XPlainEngine	**xplain**

Constructor Summary

ExplanationOWLS()
ExplanationOWLS(java.io.Writer writer)

Method Summary

void	**addCondition**(org.mindswap.owls.process.Condition condition) Método responsável por capturar a condicao e armazenar na pilha
void	**endProcess**(org.mindswap.owls.process.Process process) Metodo responsavel por capturar o processo que finalizou a execucao
void	**executionFailed**(org.mindswap.exceptions.ExecutionException e) Called when the execution fails due to an execption.
void	**executionFinished**() Called only once when the execution of the top-most process finishes.
void	**executionFinished**(org.mindswap.owls.process.Process process, org.mindswap.query.ValueMap inputs, org.mindswap.query.ValueMap outputs) Called after the execution of a process finishes.
void	**executionStarted**() Called only once when the execution of the top-most process starts
void	**executionStarted**(org.mindswap.owls.process.Process process, org.mindswap.query.ValueMap inputs) Called before the execution of a process starts.

void	**failedProcess**(org.mindswap.owls.process.Process process) Metodo responsavel por capturar o processo que falhou durante a execucao
int	**getMonitorFilter**()
void	**invokeWebService**(javax.xml.soap.SOAPMessage soap) Metodo responsável por anexar o ultimo no de prova na invocacao do WS
void	**setMonitorFilter**(int processType) Control if executionStarted and executionFinished will be called for all the processes or only for a specific type of processes.
void	**setWriter**(java.io.Writer writer)
void	**startProcess**(org.mindswap.owls.process.Process process, org.mindswap.query.ValueMap inputs) Metodo responsavel por capturar o processo que iniciou a execucao

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

xplain
public XPlainEngine **xplain**

Constructor Detail

ExplanationOWLS
public **ExplanationOWLS**()

ExplanationOWLS
public **ExplanationOWLS**(java.io.Writer writer)

Method Detail

addCondition
public void **addCondition**(org.mindswap.owls.process.Condition condition)

Description copied from interface: ExplanationInterface

Método responsável por capturar a condicao e armazenar na pilha

Specified by:

addCondition in interface ExplanationInterface

---

endProcess

public void **endProcess**(org.mindswap.owls.process.Process process)

Description copied from interface: ExplanationInterface

Metodo responsavel por capturar o processo que finalizou a execucao

Specified by:

endProcess in interface ExplanationInterface

Parameters:

process - Processo que esta finalizado a execucao

---

executionFailed

public void **executionFailed**(org.mindswap.exceptions.ExecutionException e)

Description copied from interface: org.mindswap.owls.process.execution.ProcessMonitor

Called when the execution fails due to an execption. This function is intended to be hook where the user can do something to fix the problem, i.e. ask for additional inputs. There is no such support at the moment. The execution engine will throw the exception right after this function returns.

Specified by:

executionFailed in interface org.mindswap.owls.process.execution.ProcessMonitor

---

executionFinished

public void **executionFinished**()

Description copied from interface: org.mindswap.owls.process.execution.ProcessMonitor

Called only once when the execution of the top-most process finishes. Note that if execution fails for any reason this function will not be called. Instead executionFailed() will be called.

Specified by:

executionFinished in interface org.mindswap.owls.process.execution.ProcessMonitor

---

executionFinished

public void **executionFinished**(org.mindswap.owls.process.Process process,
org.mindswap.query.ValueMap inputs,
org.mindswap.query.ValueMap outputs)

Description copied from interface: org.mindswap.owls.process.execution.ProcessMonitor

Called after the execution of a process finishes. The user has the option to modify the contents of outputs. setMonitorFilter function can be used to control if this function will be called for all the processes or only for a specific type of processes (e.g. only atomic processes).

Specified by:

executionFinished in interface org.mindswap.owls.process.execution.ProcessMonitor

---

executionStarted

public void **executionStarted**()

Description copied from interface: org.mindswap.owls.process.execution.ProcessMonitor

Called only once when the execution of the top-most process starts

Specified by:

executionStarted in interface org.mindswap.owls.process.execution.ProcessMonitor

---

executionStarted

public void **executionStarted**(org.mindswap.owls.process.Process process,
org.mindswap.query.ValueMap inputs)

Description copied from interface: org.mindswap.owls.process.execution.ProcessMonitor

Called before the execution of a process starts. The user has the option to modify the contents of the inputs. setMonitorFilter function can be used to control if this function will be called for all the processes or only for a specific type of processes (e.g. only atomic processes).

Specified by:

executionStarted in interface org.mindswap.owls.process.execution.ProcessMonitor

---

failedProcess

public void **failedProcess**(org.mindswap.owls.process.Process process)

Description copied from interface: ExplanationInterface

Metodo responsavel por capturar o processo que falhou durante a execucao

Specified by:

failedProcess in interface ExplanationInterface

Parameters:

process - Processo que ocoreu falhou durante a execucao

---

getMonitorFilter

public int **getMonitorFilter**()

Specified by:

getMonitorFilter in interface org.mindswap.owls.process.execution.ProcessMonitor

---

invokeWebService

public void **invokeWebService**(javax.xml.soap.SOAPMessage soap)

Description copied from interface: ExplanationInterface

Metodo responsável por anexar o ultimo no de prova na invocacao do WS

Specified by:

invokeWebService in interface ExplanationInterface

Parameters:

soap - Mensagem utilizada para anexar o ultimo no da prova

---

setMonitorFilter

public void **setMonitorFilter**(int processType)

Description copied from interface: org.mindswap.owls.process.execution.ProcessMonitor

Control if executionStarted and executionFinished will be called for all the processes or only for a specific type of processes. The constant values are defined in Process interface:

- Process.ANY
- Process.ATOMIC
- Process.COMPOSITE
- Process.SIMPLE

Bitwise combinations (bitwise or) of these values are also valid.

Specified by:

setMonitorFilter in interface org.mindswap.owls.process.execution.ProcessMonitor

setWriter
public void **setWriter**(java.io.Writer writer)

startProcess
public void **startProcess**(org.mindswap.owls.process.Process process,
org.mindswap.query.ValueMap inputs)

Description copied from interface: ExplanationInterface

Metodo responsavel por capturar o processo que iniciou a execucao

Specified by:

startProcess in interface ExplanationInterface

Parameters:

process - Processo que iniciou a execucao

inputs - Parametros de entrada para que o processo seja executado

br.mia.unifor.sws

Class ProofManager

java.lang.Object
  └ **br.mia.unifor.sws.ProofManager**

public class **ProofManager**
extends java.lang.Object

Constructor Summary

ProofManager()

Method Summary

void	**attachProofToSDS**(Proof proof, javax.xml.soap.SOAPMessage soap) Método responsável por anexar o ultimo no de prova gravado, na chamada do WS
Proof	**getProofAttachment**(javax.xml.soap.SOAPMessage soap) Método responsável por pegar do contexto o no de prova repassado

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ProofManager
public **ProofManager**()

Method Detail

attachProofToSDS
public void **attachProofToSDS**(Proof proof,
javax.xml.soap.SOAPMessage soap)

Método responsável por anexar o ultimo no de prova gravado, na chamada do WS

Parameters:

proof - Ultimo de no gravado

soap - Mensagem SOAP onde o nó será anexado

---

getProofAttachment
public Proof **getProofAttachment**(javax.xml.soap.SOAPMessage soap)

Método responsável por pegar do contexto o no de prova repassado

Parameters:

soap - Mensagem SOAP onde o nó foi anexado

Returns:

Ultimo no de gravado, que foi repassado como anexo

Overview **Package** **Class** **Use** **Tree** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES** **All Classes**

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Overview **Package** **Class** **Use** **Tree** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES** **All Classes**

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

br.mia.unifor.sws

Class WSRegistrarProvas

java.lang.Object
  └ **br.mia.unifor.sws.WSRegistrarProvas**

---

public class **WSRegistrarProvas**
extends java.lang.Object

Constructor Summary

WSRegistrarProvas()

Method Summary

java.lang.String	**recorderProof**(Proof proof) Metodo responsavel por receber as provas repassadas pelos SW

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

WSRegistrarProvas
public **WSRegistrarProvas**()

Method Detail

recorderProof
public java.lang.String **recorderProof**(Proof proof)
Metodo responsavel por receber as provas repassadas pelos SW
Parameters:
proof - Prova que deve ser gravada
Returns:

Overview **Package** **Class** **Use** **Tree** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES** **All Classes**
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Overview **Package** **Class** **Use** **Tree** **Index** **Help**

PREV CLASS NEXT CLASS **FRAMES** **NO FRAMES** **All Classes**
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

br.mia.unifor.sws

Class *XPlainS*Engine

java.lang.Object
  └ **br.mia.unifor.sws.*XPlainS*Engine**

public class ***XPlainS*Engine**
extends java.lang.Object

Constructor Summary

***XPlainS*Engine**()

Method Summary

void	**addCondition**(org.mindswap.owls.process.Condition condition) Adiciona a condicao na pilha de condicoes

void	**generateLocalProof**(org.mindswap.owls.process.Process process) Registra as regras referentes a chamada da explicacao
void	**generateProofInIWRegistrar**() Método que acessa o IWRegistrar para registrar a prova em PML
void	**generateProofInWS**(Proof proof) Registra as provas que foram gravadas localmente no SW que iniciou o processo.
br.mia.unifor.util.StackCondition	**getStackCondition**() Método que retorna a pilha de condicoes

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

*XPlainS*Engine
public ***XPlainS*Engine**()

Method Detail

addCondition
public void **addCondition**(org.mindswap.owls.process.Condition condition)
Adiciona a condicao na pilha de condicoes
Parameters:
condition - Condicao a ser armazenada na pilha de condicoes

---

generateLocalProof
public void **generateLocalProof**(org.mindswap.owls.process.Process process)
Registra as regras referentes a chamada da explicacao
Parameters:
process - Processo que chamou a explicacao

---

generateProofInIWRegistrar
public void **generateProofInIWRegistrar**()
Método que acessa o IWRegistrar para registrar a prova em PML

---

generateProofInWS
public void **generateProofInWS**(Proof proof)
Registra as provas que foram gravadas localmente no SW que iniciou o processo.
Parameters:
proof - - Prova Gravada Localmente

---

getStackCondition

public br.mia.unifor.util.StackCondition **getStackCondition**()

Método que retorna a pilha de condicoes

Returns:

Retorna a pilha de condicoes

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo