Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Informática
Tecnologia em Desenvolvimento de Sistemas Distribuídos

Maurício Chicalski

Protótipo de um Sistema Web para Desenvolvimento Colaborativo de Ontologias

Trabalho de Conclusão de Curso

CURITIBA
2013

Maurício Chicalski

Protótipo de um Sistema Web para Desenvolvimento Colaborativo de Ontologias

Trabalho de Conclusão do Curso de Tecnologia em desenvolvimento de sistemas distribuídos, apresentado à UTFPR como requisito parcial para obtenção do título de tecnólogo em desenvolvimento de sistemas distribuídos.

Orientador: César A. Tacla

Curitiba
2013

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CAMIO: Collaborative Argumentation Merging Individual Ontologies.

IA: Inteligência Artificial.

OWL: Web Ontology Language.

DL: Description Logics.

IC: Identity Criteria.

UC: Unity Criteria.

LOD: Linked Open Data.

Oi: Ontologia individual.

Oc: Ontologia colaborativa.

ns: namespace (foi usado para generalizar namespaces quaisquer).

URI: Uniform Resource Identifier.

UML: Unified Modeling Language.

API: Application Programming Interface.

EE: Enterprise Edition.

JSR: Java Specification Request.

CDI: Contexts and Dependency Injection.

JSF: JavaServer Faces.

JPA: Java Persistence API.

EJB: Enterprise JavaBeans.

DI: Dependency Injection.

JTA: Java Transaction API.

SSE: Server Side Events.

DOM: Document Object Model.

HTML: HyperText Markup Language.

XHTML: eXtensible HTML.

ODM: Ontology Definition Metamodel.

MDA: Model Driven Architecture.

EMF: Eclipse Modeling Framework.

EMT: Eclipse Modeling Tools.

XML: eXtensible Markup Language.

RDF: Resource Description Framework.

RDFS: RDF-Schema.

DAO: Data Access Object.

EL: Expression Language.

HTTP: Hypertext Transfer Protocol.

AJAX: Asynchronous JavaScript and XML.

JS: JavaScript.

CSS: Cascading Style Sheets.

SUMÁRIO

1 Introdução

Este trabalho trata da implementação de um protótipo para um sistema que contempla o método CAMIO (Collaborative Argumentation Merging Individual Ontologies) ou argumentação colaborativa para a combinação de ontologias individuais.

1.1 Formulação do Problema de Pesquisa

Uma das especializações da computação trata sobre trabalho colaborativo suportado por computador que, considera principalmente o local onde os participantes estão e o tempo em que ocorrem as ações de cada participante. Durante a implementação de um sistema computacional que suporte uma determinada atividade colaborativa, os problemas são derivados do tipo de atividade colaborativa a ser suportada pelo sistema. As soluções devem considerar paradigmas de sistemas colaborativos e distribuídos.

Embora o desenvolvimento de ontologias disponha de excelentes ferramentas e métodos para uso individual, há uma crescente necessidade de métodos e padrões que explorem o desenvolvimento de modo colaborativo.

1.2 Objetivos

Desenvolver um sistema Web para apoiar o desenvolvimento de ontologias, considerando especialmente a colaboração entre usuários e a validação da estrutura taxonômica produzida.

1.2.1 Ontologia Colaborativa

Realizar a combinação de diversas ontologias individuais pré-existentes (dentro de um domínio comum) em uma nova ontologia chamada colaborativa.

O sistema tem como entrada as ontologias individuais. Então com a participação dos usuários em um processo de discussão e votação, define-se a ontologia colaborativa. Esta nova ontologia deve representar uma visão geral e consensual entre os participantes.

1.2.2 Estrutura Taxonômica Válida

Auxiliar o processo de discussão e votação, de modo que a estrutura taxonômica criada esteja correta com base na metodologia OntoClean (GUARINO, WELTY 2004).

1.3 Motivação e Justificativa

Na UTFPR há um grupo de pesquisa denominado MEMENTO que, entre outros objetos de estudo, realiza trabalhos sobre o desenvolvimento e a aplicação de ontologias.

O grupo é formado por professores e estudantes e, ao longo da graduação, o autor participou de algumas atividades dentro do grupo e pôde constatar:

- O desenvolvimento de ontologias demanda a participação conjunta tanto de especialistas na criação de ontologias, quanto de especialistas nos domínios a serem representados.
- O processo de desenvolvimento também pode acabar utilizando muito tempo e recursos, tornando-se um processo complicado, caro e demorado especialmente se os envolvidos necessitarem estar no mesmo local ao mesmo tempo.

Entre os trabalhos recentes do grupo, está a elaboração de um método aplicado ao desenvolvimento colaborativo de ontologias, chamado Folkoncept (FREDDO 2010). O método foi formulado, e precisava ser implementado em um sistema para ser experimentado. Num segundo momento, este método foi melhorado e, novamente, precisava ser implementado para ser avaliado em experimentos.

1.4 Estrutura do Trabalho

O trabalho apresenta, no segundo capítulo, uma breve revisão dos principais conceitos e termos que são a base fundamental do sistema. Define o conceito ontologia, apresenta os fundamentos da metodologia OntoClean, descreve o método Folkoncept e cita trabalhos que envolvem colaboração na Linked Open Data (LOD).

No terceiro capítulo é feita uma pequena revisão sobre o método CAMIO e apresentada a forma que o método influenciou a modelagem geral do sistema.

O quarto capítulo apresenta as tecnologias empregadas no desenvolvimento do sistema e a finalidade de cada uma.

O quinto capítulo mostra os resultados, especificamente a documentação gerada durante o desenvolvimento do protótipo do sistema.

O sexto capítulo tem uma breve discussão sobre o decorrer do desenvolvimento.

O sétimo capítulo encerra o trabalho com as conclusões obtidas e aponta o que poderia ser considerado em uma eventual transição do status de protótipo dado ao sistema desenvolvido.

2 Levantamento Bibliográfico e Estado da Arte

Este capítulo apresenta os resumos e as interpretações das referências consultadas durante o projeto.

2.1 Ontologia

De acordo com (GUARINO 1998), em seu uso mais prevalente na IA (Inteligência Artificial), uma ontologia se refere a um artefato de engenharia, constituído por um vocabulário específico usado para descrever uma certa realidade, mais um conjunto de pressupostos explícitos sobre o significado pretendido das palavras do vocabulário. Esse conjunto de pressupostos geralmente tem a forma de uma teoria lógica de primeira-ordem, onde as palavras do vocabulário aparecem como nomes de predicados unários ou binários, respectivamente chamados de conceitos e relações.

A linguagem OWL-DL (Web Ontology Language – Description Logics) tem sido usada para representar Ontologias e é baseada em um subconjunto da lógica de primeira-ordem – que é a Description Logics (DL) ou Lógica de Descrições.

Uma ontologia expressa em DL possui uma terminologia (conceitos e relações) chamada TBOX e uma série de asserções (com base na terminologia) que estão na ABOX (ver figura 1).

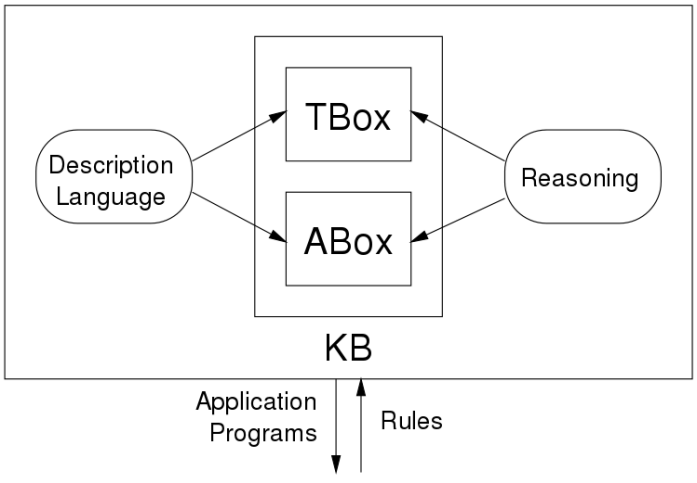

Figura 1. Arquitetura de um sistema de representação de conhecimento baseado em Lógica de Descrições (BADDER, NUTT 2003).

2.2 OntoClean

A metodologia OntoClean é baseada em noções provenientes da filosofia tais como essência, identidade, unidade e dependência.

Para descrever a metodologia OntoClean, utiliza-se o termo propriedade. Segundo os autores da metodologia (GUARINO, WELTY 2004), uma propriedade equivale a um predicado unário na lógica de primeira-ordem. Dado um mundo possível, uma classe ou conceito corresponde às instâncias que exibem tal propriedade.

As noções de essência, identidade, unidade e dependência são chamadas de metapropriedades ou propriedades de propriedades. As metapropriedades impõem restrições hierárquicas às propriedades.

Considere que não há um critério de que uma determinada propriedade deva possuir determinadas atribuições às metapropriedades como certas ou erradas. Segundo (WELTY 2006), se dois analistas diferentes adotam atribuições conflitantes às metapropriedades, então eles têm diferentes interpretações. É fundamental expôr essas diferenças pois ontologias devem capturar um "entendimento comum" ou "significados compartilhados" dos termos.

O principal objetivo da metodologia é permitir a avaliação das consequências lógicas das escolhas feitas durante a modelagem taxonômica em uma ontologia.

2.2.1 Essência

A noção de essência e rigidez é definida pela ocorrência da propriedade nas mesmas instâncias em outros mundos possíveis. Pode ser rígida (+R), ou não rígida, sendo, semirrígida (-R) ou antirrígida (~R):

- **+R:** propriedades essenciais para todas as instâncias;
- **-R:** propriedades não essenciais para algumas das instâncias, não subsumem propriedades +R**;**
- **~R:** propriedades não essenciais para todas as suas instâncias, subsumem apenas propriedades ~R.

2.2.2 Identidade

A noção de identidade é definida pela presença (+I) ou ausência (-I) de um Identity Criteria (IC) nas instâncias, permitindo identificá-las e diferenciá-las. O IC pode identificar a instância por um momento ao longo do tempo (sincrônico) ou ao longo de todo o tempo (diacrônico). Há também a separação entre propriedades que fornecem (+O) e não fornecem (-O) um IC:

- **+I:** propriedades que carregam um IC, não subsumem propriedades -I. As instâncias destas propriedades podem ser enumeradas, correspondem a substantivos;
- **-I:** propriedades que não carregam um IC, são também -O. As instâncias destas propriedades não podem ser enumeradas, geralmente correspondem a adjetivos;
- **+O:** propriedades que fornecem IC, também são obrigatoriamente +R e +I.
- **-O:** propriedades que não fornecem um IC.

2.2.3 Unidade

A noção de unidade é definida pela capacidade de delimitar as partes de uma instância do resto do mundo por meio de uma relação unificadora que as une. São classificadas quanto ao tipo do Unity Criteria (UC), podendo ser unitária (+U), semiunitária (-U) ou antiunitária (~U).

- **+U:** propriedades delimitáveis sob um critério comum à todas instâncias, subsumem apenas propriedades +U.
- **-U:** propriedades delimitáveis sob diferentes critérios para diferentes instâncias.
- **~U:** propriedades com instâncias não delimitáveis, subsumem apenas propriedades ~U.

2.2.4 Dependência

A noção de dependência é definida pela dependência ou não de outra propriedade. A dependência pode ser intrínseca (interna) ou extrínseca (externa). Apenas as dependências extrínsecas são consideradas, pois as

dependências intrínsecas representam partes ou constituintes próprios de uma instância.

- **+D:** propriedades que dependem de outra propriedade, subsumem apenas propriedades +D. As instâncias destas propriedades dependem de instâncias de outras propriedades.
- **-D:** propriedades que não dependem de outra propriedade.

2.3 Folkoncept

O Folkoncept (FREDDO 2010) é um método de suporte à modelagem conceitual de ontologias a partir da aquisição de conhecimentos de folksonomias.

O método tem o objetivo de reduzir o gargalo na aquisição de conhecimentos, utilizando técnicas de aprendizado de ontologias a partir de folksonomias.

As folksonomias são um conjunto de palavras ou etiquetas (tags) que anotam algum recurso. O Folkoncept enriquece essas palavras através da recuperação de informações da WordNet, uma ontologia que descreve algumas relações entre palavras.

O método auxilia na criação de novos elementos na ontologia a partir do conjunto de tags enriquecidas. Também aplica transparentemente a metodologia OntoClean, para avaliação da estrutura taxonômica.

A colaboração entre usuários pode ocorrer na fase de criação da folksonomia, porém na fase de criação dos elementos na ontologia o método é aplicado individualmente, resultando em uma ontologia por usuário.

2.4. LOD

A *Linked Open Data* (LOD), ou dados abertos ligados, interliga uma crescente quantidade de dados e ontologias. O termo Linked Data refere-se a um conjunto de melhores práticas para publicar e conectar dados estruturados na Web.

Para (BIZER, et al. 2009), a adoção das melhores práticas de Dados Ligados levou à extensão da Web com um espaço global de dados conectando dados de diversos domínios, tais como pessoas, empresas,

livros, publicações científicas, filmes, música, programas de rádio e televisão, genes, proteínas, drogas e ensaios clínicos, comunidades online, dados estatísticos e científicos, e opiniões. Esta Web de dados permite novos tipos de aplicações.

A LOD é formada por diversas fontes de dados. Os projetos Freebase e DBpedia, se destacam como fontes de dados mais gerais. A figura 2 é um retrato da LOD em março de 2009. Outro retrato de setembro de 2011 é consideravelmente maior e, consolida a base DBpedia como um *hub* centralizador das interligações.

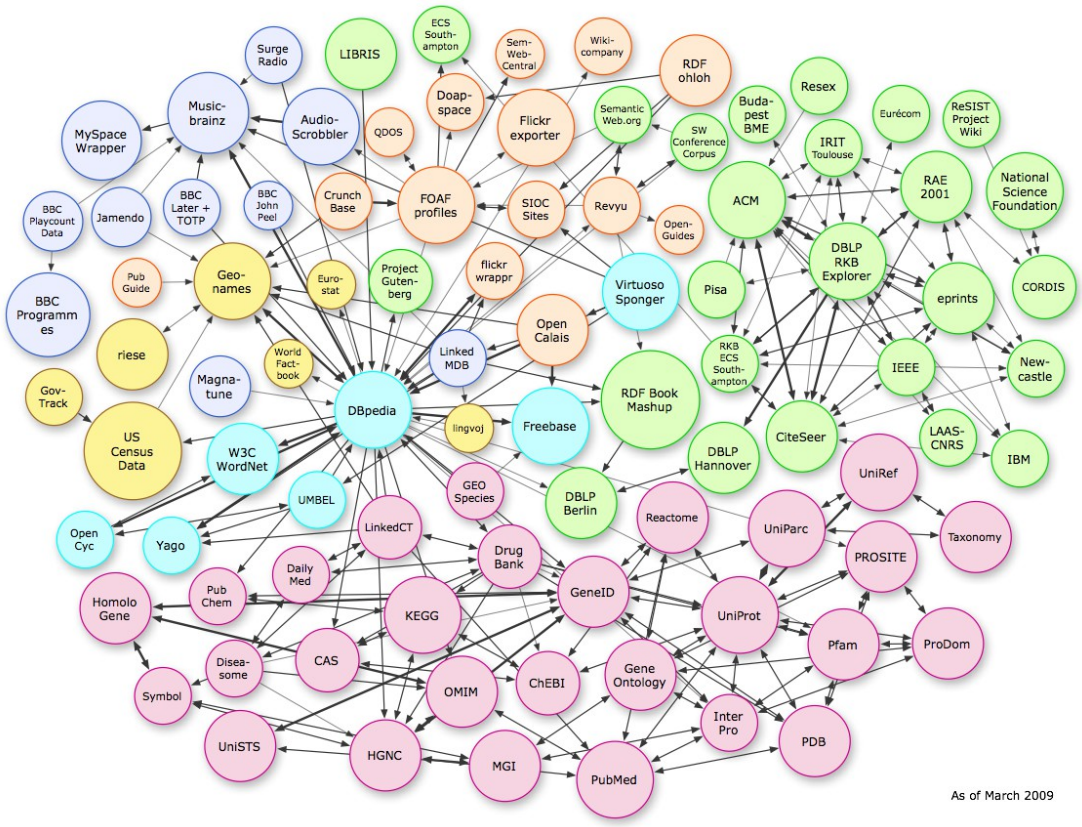

Figura 2. Resumo dos conjuntos de dados publicados na LOD e seus relacionamentos de interligação (BIZER, et al. 2009).

O Freebase é um banco de dados de informação estruturada geral, criado e editado colaborativamente.

As características do Freebase são projetadas para facilitar alta "densidade colaborativa" entre seus usuários na organização, representação

e integração de grandes conjuntos de dados diferentes (BOLLACKER, et al. 2007). O Freebase também fornece interfaces de acesso para agentes (pessoas e agentes de software) trabalharem de forma assíncrona.

A Wikipédia é uma importante fonte de informações aberta. É uma enciclopédia livre, multilíngue, editada colaborativamente, por contribuidores ao redor do mundo, gerenciada pela WikiMedia Foundation como consta em (WIKIPEDIA 2013).

As informações de muitos artigos não são tão confiáveis, mas boa parte dos artigos contém informações relativamente estruturadas, algumas destas estruturas são as chamadas *infoboxes*.

O DBpedia é um projeto aberto da comunidade (DBPEDIA 2013) cujo objetivo é extrair informação estruturada da Wikipédia e tornar esta informação disponível na Web.

Mapeamentos (*mappings)* ligam *infoboxes* de artigos Wikipedia ao TBOX da ontologia DBpedia, de forma que o ABOX seja extraído automaticamente das páginas. Os mapeamentos e o TBOX da ontologia são editados manual e colaborativamente por usuários, através de *wikipages* semelhante ao modelo de colaboração da Wikipedia.

De certa forma pode-se dizer que a ontologia extraída da Wikipedia conta com a colaboração indireta dos usuários da Wikipedia, e a colaboração direta dos usuários do projeto DBpedia.

3 Desenvolvimento Colaborativo de Ontologias

Este capítulo apresenta a descrição geral do método CAMIO (DALL'AGNOL 2013) e respectivas influências na modelagem do sistema. A seção 3.1 descreve o método em sua essência, enquanto que nos demais o método está adaptado às necessidades e possibilidades de implementação do sistema.

3.1 Argumentação colaborativa para combinação de ontologias

A figura 3 resume o processo para combinação de ontologias individuais proposto pelo método CAMIO.

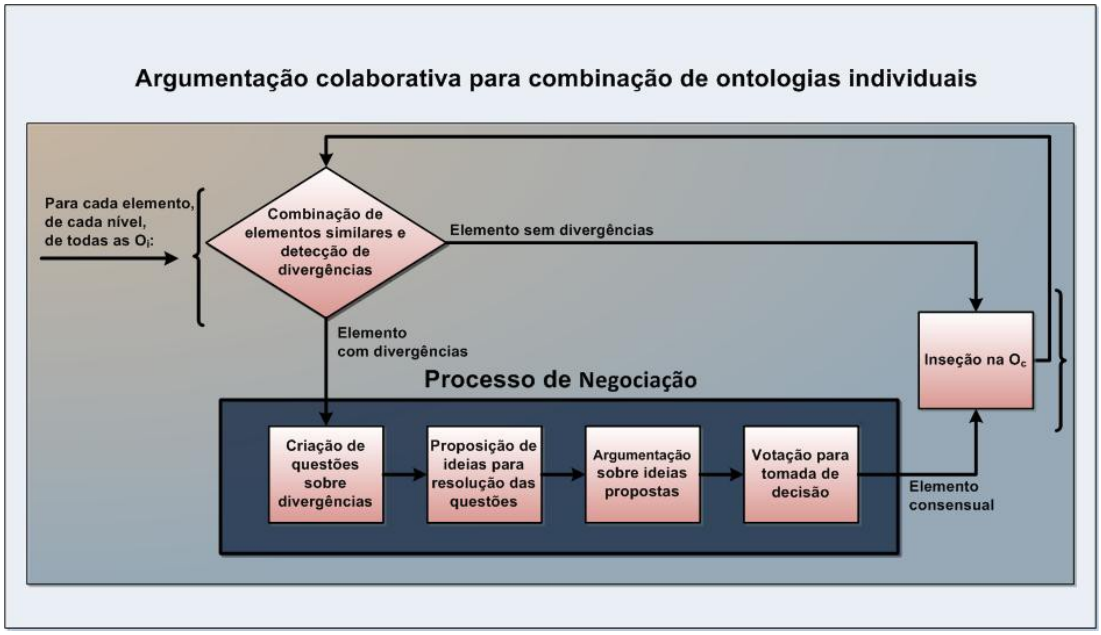

Figura 3. Argumentação colaborativa para combinação de ontologias individuais (DALL'AGNOL 2013).

A combinação tem início após a reunião das ontologias individuais (Oi). Para cada elemento de uma Oi, verifica-se se o mesmo elemento está presente nas demais Oi. O elemento que possuir as mesmas características (taxonômica e OntoClean) em todas Oi, é incluído diretamente na ontologia colaborativa (Oc). Caso exista alguma divergência na taxonomia ou nas metapropriedades OntoClean, o elemento passa por um processo de negociação, onde os participantes do grupo devem argumentar e chegar em

um consenso sobre qual das alternativas a maioria do grupo considera correta. A escolha do grupo é então incluída na Oc.

3.2 Ontologia de Argumentação

A ontologia de argumentação se resume em um conjunto de entidades que representam o estado do processo de combinação, discussão e votação no sistema. Um arquivo OWL-DL representando a ontologia colaborativa pode ser criado a partir de um algoritmo que lê o estado destas entidades. Este arquivo OWL-DL é o resultado do processo, podendo ser importado por outras aplicações.

- **Recurso** - representa um recurso da ontologia, é um elemento que poderia ser um conceito, relação, restrição etc. Cada tipo de recurso possui suas respectivas questões. No contexto de um fato, é um sujeito. Ex: ns:Pessoa, ns:Cidade, ns:cidadeNatal.
- **Questão** - representa uma relação entre recursos ou entre recurso-valor, dependendo do tipo da questão. No contexto de um fato é um predicado. Ex: rdfs:subClassOf, rdfs:domain, oc:unity.
- **Ideia** - cada ideia representa uma entre várias respostas possíveis a uma questão. A ideia mais votada é a resposta dada à questão. No contexto de um fato, é um objeto. Ex: ns:Animal, “+R”, “-R”.
- **Voto** - representa o voto de um usuário para uma ideia entre as ideias disponíveis de uma questão.
- **Grupo** - cada grupo representa um processo independente de colaboração.
- **Ator** - usuários participantes, podem ser do tipo usuário, administrador ou sistema.
- **Template** - frases com variáveis a serem substituídas, os textos descrevem questões e ideias. Ideias são concatenadas para explicar as violações de restrição OntoClean.
- **Argumento** - relacionam-se a uma ideia, podem ser criados por qualquer tipo de ator, o sistema gera argumentos a partir de templates.
- **Ontologia** - representa cada ontologia importada.

- **Seta** – surgiu como uma classe de ajuda para auxiliar a interface do usuário. Cada seta equivale a um *statement* com o predicado rdfs:subClassOf.

3.3 Combinação de Ontologias

Em uma primeira implementação, para participar de um grupo cada usuário necessitava de exatamente uma ontologia, sendo que todos os usuários precisavam se reunir e carregar suas respectivas ontologias individuais, para então disparar a combinação e dar início às discussões.

O algoritmo de combinação instancia os elementos na ontologia de argumentação, a partir da leitura de todas as ontologias individuais (arquivos OWL) e para cada conceito que existir:

- seleciona ou cria na ontologia de argumentação um recurso r1 com a mesma URI (Uniform Resource Identifier) do conceito;
- seleciona ou cria uma questão taxonômca q1, para o recurso atual r1;
- seleciona ou cria na ontologia de argumentação um recurso r2 com a mesma URI do superconceito do conceito atual r1;
- seleciona ou cria na ontologia de argumentação uma ideia i1 [r1 subClassOf r2] e relaciona a ideia i1 à questão q1;
- cria um voto v1 para a ideia i1, este voto está relacionado ao usuário u1 da ontologia individual oi1;

Dentro deste processo também ocorre a importação das metapropriedades OntoClean. Após a execução, as ontologias individuais estão representadas na ontologia de argumentação.

Para cada ideia na ontologia de argumentação, outro algoritmo verifica se o número de votos da ideia é igual ao número de ontologias individuais. Se sim, houve consenso e atribui o status da questão para consenso. Se não, remove os votos e atribui o status da questão para discussão.

3.4 Importação de Ontologias

A principal mudança esperada em relação à combinação das ontologias é a independência do grupo em relação aos usuários e a possibilidade de adicionar novas ontologias após o início das discussões.

O processo de combinação passa a se chamar importação, pois recebe como entrada apenas uma ontologia individual por execução.

Durante a importação, os usuários não ficam mais relacionados ao voto da ideia importada, e cada ideia recebe uma propriedade extra chamada contagem de importação. Esta alteração permite que um único usuário combine várias ontologias, como único paraticipante de um grupo. Posteriormente esta propriedade contagem de importação é tratada como votos do sistema.

A ausência de um elemento em uma ontologia individual (porém existente na ontologia de argumentação) pode ser considerada pelo sistema como um voto para remover o elemento. Isto pode permitir uma rápida extração dos conceitos com mais interseções entre diferentes ontologias.

Mas se um único usuário importou n ontologias, existe a possibilidade de uma ideia possuir n votos do sistema, e seriam necessários n votos de n usuários para pelo menos empatar com o sistema. Como todo usuário tem direito a somente um voto por questão, isto implicaria que os votos do sistema poderiam prevalecer sobre os votos dos usuários. Para contornar esta situação considera-se que entre as ideias de uma questão, todos os votos das importações valem ponderadamente 1.

Esta diferenciação torna possível carregar novas ontologias e que novos usuários possam entrar para um grupo já em andamento.

3.5 Fórum de Discussões

Inicialmente a ontologia de argumentação ficava disposta em uma página, análoga e esteticamente semelhante a um fórum de discussões:

- grupos → assuntos do fórum;
- recursos → tópicos por assunto;
- questões → subtópicos por tópico*;*
- ideias → opções de votação por subtópico*;*
- argumentos → mensagens por opção de votação.

Um sistema semelhante aos *hot-topics* foi aplicado na tentativa de guiar os participantes “simultaneamente” às mesmas questões. O uso de *hot-topics* é bem comum em fóruns de discussões. O objetivo é destacar certos tópicos individualmente por usuário, até que ocorra um determinado evento por parte deste usuário, cessando o destaque, também individualmente.

3.6 Diagrama

Alguns testes constataram que à medida que as discussões desciam os níveis na árvore taxonômica, ficava difícil enxergar o contexto da discussão e das ideias em votação, pois os conceitos ficavam dispostos como registros em uma “tabela de tópicos”. Mesmo assim a analogia a um fórum de discussões é suficientemente boa para a manipulação de toda a ontologia de argumentação.

Foi experimentado posicionar cada registro da tabela de tópicos independentemente na tela do usuário, e desenhar setas entre estes registros, utilizando as ideias taxonômicas como fonte de setas. Desta forma surgiu um “diagrama hierárquico” a partir dos registros da tabela de tópicos. O objetivo do diagrama é permitir a visualização do contexto de cada conceito em discussão, mantendo a ideia inicial do fórum de discussões.

4 Metodologia

O processo de desenvolvimento foi em paralelo com a tese (DALL'AGNOL 2013) que elaborou o método CAMIO.

O ciclo de vida do projeto foi em espiral. O marco ou a conclusão de um ciclo ocorria com testes do sistema durante a experimentação do método. Ao final de cada experimento, os participantes respondiam um formulário de avaliação sobre o método, incluindo críticas e sugestões ao sistema. Desta forma, obtinham-se retornos dos usuários que geravam revisões dos requisitos funcionais, dando início ao novo ciclo. No total ocorreram 5 ciclos completos de desenvolvimento.

O planejamento do processo de desenvolvimento foi feito a partir de casos de uso. As funcionalidades implementadas foram documentadas em diagramas de classe e de sequência seguindo o padrão UML (Unified Modeling Language). Parte da documentação está no capítulo Resultados.

A tabela 1 mostra quantas vezes os experimentos foram repetidos e em que situações ocorreram.

Tabela 1. Experimentos do Sistema.

Experimento	Mês/Ano	Origem dos Participantes	Grupos	Participantes por Grupo
1	03/2012	Ciência da Computação/ Unicentro	11	3
2	08/2012	Ciência da Computação/ Unicentro	2	3
3	10/2012	Engenharia da Computação/ UTFPR	8	3
4	10/2012	Ciência da Computação/ Unicentro	7	3
5	12/2012	Disciplina Ontologias/ CPGEI UTFPR	2	3, 4

4.1 Tecnologias

A linguagem Java foi escolhida para o desenvolvimento do núcleo do sistema. A linguagem JavaScript foi utilizada para complementar as funcionalidades da interface do usuário.

4.1.1 Java Enterprise Edition

O Java EE 6 é uma extensa especificação definida através da JSR 316 (Java Specification Request), esta inclui diversas outras especificações, cobrindo uma incrível variedade de tópicos.

A figura 4 demonstra a arquitetura Java EE 6. Os retângulos são os *containers,* eles são os ambientes de execução Java EE, são eles quem disponibilizam os serviços requeridos pelos componentes da aplicação. Em cada *container* estão listados os serviços que cada um disponibiliza.

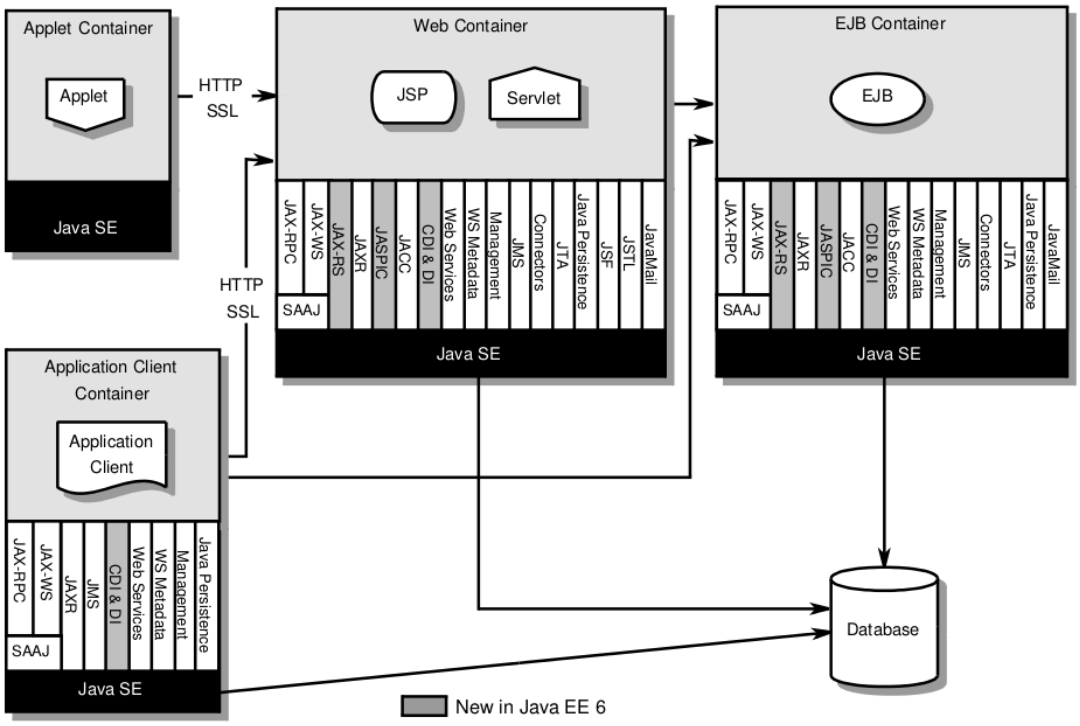

Figura 4. Arquitetura Java EE 6 (JSR 316).

A tabela 2 cita os principais serviços utilizados no projeto. Maiores informações destes ou demais serviços e especificações podem ser obtidas através de suas respectivas JSRs.

Tabela 2. Principais serviços Java EE utilizados.

Sigla	Nome	JSR
CDI	Contexts and Dependency Injection for Java (Web Beans 1.0)	299
JSF	JavaServer Faces 2.0	314
JPA	Java Persistence 2.0	317
EJB	Enterprise JavaBeans 3.1	318
DI	Dependency Injection for Java 1.0	330
JTA	Java Transaction API 1.1	907

O uso de serviços dos containers Web e EJB requerem o uso de servidores de aplicação. Inicialmente utilizava-se o servidor Glassfish 3.1. Algumas razões levaram o projeto a ser reescrito e a nova versão acabou migrando para o servidor TomCat que dispõe apenas do container Web, mas em conjunto ao projeto TomEE (TomCat Enterprise Edition), o container EJB também é adicionado. Embora tenha ocorrido a migração, o projeto deveria (com poucas reconfigurações) estar apto a rodar em qualquer servidor de aplicação que suporte os containers Web e EJB.

O PrimeFaces é um entre vários frameworks de componentes JSF. Um dos seus componentes que necessita ser citado é o PrimePush, que em conjunto com o servidor Atmosphere, torna possível o uso de Server Side Events (SSE), onde a aplicação rodando no servidor pode a qualquer momento enviar eventos a clientes (páginas web). O uso de SSEs foi considerado um requisito funcional após alguns experimentos iniciais, pois são extremamente necessários quando existem inúmeros usuários atualizando simultaneamente as mesmas entidades compartilhadas.

A API Java Persistence torna transparente o acesso a sistemas gerenciadores de banco de dados. A JPQL (Java Persistence Query Language) mantém a transparência entre as sintaxes e implementações particulares da linguagem SQL (Structured Query Language). O sistema gerenciador de banco de dados tem sido o MySQL.

4.1.2 JavaScript

JavaScript é uma linguagem interpretada, amplamente suportada por navegadores Web.

A primeira biblioteca incorporada ao projeto foi a JQuery, que apresenta muitas facilidades em relação a manipulação de elementos DOM (Document Object Model), disponibilizando diversas funcionalidades implementadas em JavaScript.

Em seguida adicionou-se dependência à biblioteca JsPlumb. Ela permite gerenciar conexões (desenháveis na página) entre elementos DOM de páginas HTML.

A última biblioteca JavaScript importada pelo projeto foi a Dagre, responsável por executar o “directed graph layout” no lado do cliente, ou seja, executa o posicionamento dos recursos no navegador do usuário.

4.1.3 ODM

UML é uma linguagem padrão de notação de projetos. A linguagem pode ser facilmente estendida através de *profiles* UML.

A ODM (Ontology Definition Metamodel) é uma especificação para que a MDA (Model Driven Architecture) geralmente aplicada em engenharia de sistemas, possa ser aplicada a engenharia de ontologias. A ODM disponibiliza um *profile* que estende a UML.

O projeto analisou a possibilidade de utilizar modelos UML para representação das ontologias no sistema, através das APIs Java para UML EMF (Eclipse Modeling Framework) e EMT (Eclipse Modeling Tools) em conjunto ao profile ODM. As APIs UML foram descartadas, mas no sistema a representação gráfica do diagrama do fórum é inspirada nas notações visuais propostas pela especificação ODM.

A figura 5 mostra a proposta de uma notação visual para representar uma redefinição de propriedade, neste caso uma restrição para a imagem da relação.

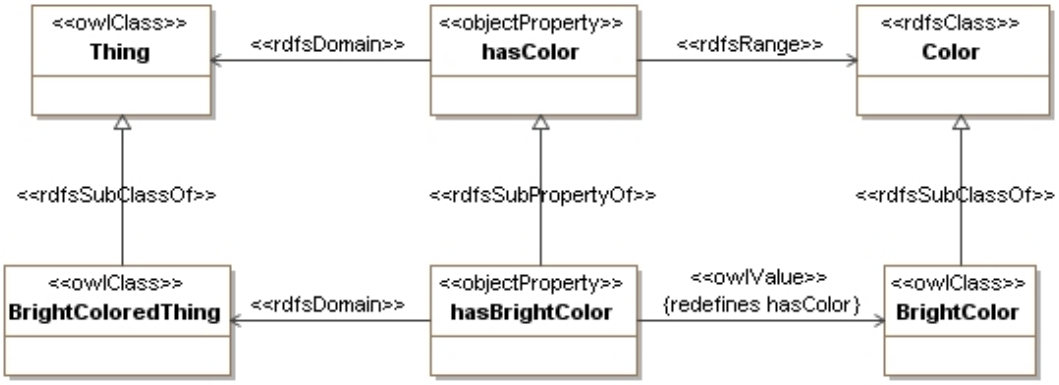

Figura 5. Redefinição de propriedade para owl:allValuesFrom (ODM 2009).

4.1.4 APIs Ontologia

Apache Jena™ é um framework Java para a construção de aplicações para a Web Semântica. Jena fornece uma coleção de ferramentas e bibliotecas. O projeto utilizou principalmente a API para ler, processar e escrever dados RDF (Resource Description Framework) em XML (eXtensible Markup Language), e também a API ontologia para manipular ontologias OWL e RDFS (RDF-Schema).

A OWL API, é outra API Java para Web Semântica, oferece recursos semelhantes a Apache Jena, concentrada na OWL2. A OWL API foi escolhida para substituir a Jena, devido a alguns conflitos de versão entre uma dependência chamada *xerces*, utilizada pela biblioteca Jena e pelos servidores de aplicação testados.

Para a validação das restrições OntoClean, foram testados alguns *reasoners* através da metodologia e ferramenta proposta por (WELTY 2006), porém seria necessário interpretar as respostas dadas pelo *reasoner,* e quando haviam várias violações simultâneas, nem todas eram descritas na resposta, o que inviabilizaria o objetivo do sistema de identificar e argumentar todas as violações num dado momento. Então optou-se em executar a identificação de violações através de um algoritmo descrito na seção *5.3.7 Validar Restrições OntoClean.*

5 Resultados

Este capítulo apresenta a documentação UML, primeiramente o diagrama de casos de uso (ver figura 6), em seguida, os detalhes de cada caso de uso (tabelas 3 a 10). Os diagramas de classes que foram gerados a partir do código-fonte existente (figuras 7 a 10), e por fim as telas do sistema que implementam os casos de uso (figuras 11 a 14).

5.1 Casos de Uso

Os principais casos de uso estão na figura 6, os atores administrador e sistema são especializações do ator usuário, cada um adiciona um caso de uso específico.

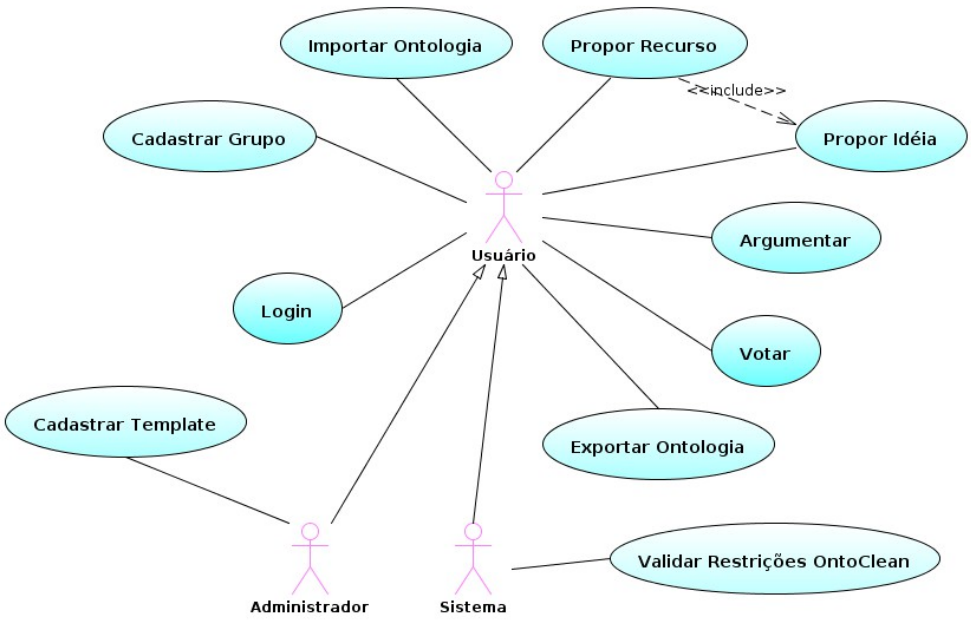

Figura 6. Casos de uso do sistema.

Tabela 3. Detalhes do caso de uso Login.

Propriedade	Valor
Nome	Login
Ator	Usuário
Descrição	Usuário efetua *login* no sistema através da página web
Pré-Condições	Usuário cadastrado
Pós-Condições	Usuário logado
Fluxo Básico	Criação da sessão para o usuário
Fluxo Alternativo	Mensagem de *login* inválido

Tabela 4. Detalhes do caso de uso Cadastrar Grupo.

Propriedade	Valor
Nome	Cadastrar Grupo
Ator	Usuário
Descrição	Usuário cadastra um grupo no sistema através da página grupos
Pré-Condições	Usuário logado
Pós-Condições	Grupo cadastrado com respectiva ontologia de argumentação
Fluxo Básico	Criação do grupo

Tabela 5. Detalhes do caso de uso Importar Ontologia.

Propriedade	Valor
Nome	Importar Ontologia
Ator	Usuário
Descrição	Usuário importa uma ontologia na ontologia de argumentação através da página do grupo
Pré-Condições	Grupo cadastrado
Pós-Condições	Ontologia importada à ontologia de argumentação
Fluxo Básico	Importação da ontologia
Fluxo Alternativo	Mensagem de ontologia inválida

Tabela 6. Detalhes do caso de uso Propor Recurso.

Propriedade	Valor
Nome	Propor Recurso
Ator	Usuário
Descrição	Usuário propõe um novo recurso para a ontologia de argumentação através da página do grupo
Pré-Condições	Grupo cadastrado
Pós-Condições	Novo recurso criado
Fluxo Básico	Criação do recurso na ontologia de argumentação, incluindo propor uma ideia taxonômica referenciando o novo recurso

Tabela 7. Detalhes do caso de uso Propor Ideia.

Propriedade	Valor
Nome	Propor Ideia
Ator	Usuário
Descrição	Usuário propõe uma nova ideia para um recurso na ontologia de argumentação a partir de um recurso da página do grupo
Pré-Condições	Recurso existente na ontologia de argumentação
Pós-Condições	Nova ideia criada
Fluxo Básico	Criação da ideia proposta referenciando o recurso na ontologia de argumentação

Tabela 8. Detalhes do caso de uso Argumentar.

Propriedade	Valor
Nome	Argumentar
Ator	Usuário
Descrição	Usuário argumenta uma ideia a partir das ideias de um recurso
Pré-Condições	Recurso e ideia existentes na ontologia de argumentação
Pós-Condições	Novo argumento criado
Fluxo Básico	Criação do argumento para a ideia

Tabela 9. Detalhes do caso de uso Votar.

Propriedade	Valor
Nome	Votar
Ator	Usuário
Descrição	Usuário escolhe uma ideia entre todas as ideias de um recurso
Pré-Condições	Recurso e ideia existentes na ontologia de argumentação
Pós-Condições	Voto contabilizado
Fluxo Básico	Contabilizar voto à ideia escolhida

Tabela 10. Detalhes do caso de uso Exportar Ontologia.

Propriedade	Valor
Nome	Exportar Ontologia
Ator	Usuário
Descrição	Usuário exporta uma ontologia contendo as ideias vencedoras na ontologia de argumentação
Pré-Condições	Recurso existente na ontologia de argumentação
Pós-Condições	Arquivo OWL exportado
Fluxo Básico	Exportação da ontologia colaborativa

5.2 Diagramas de Classes

Os diagramas de classes estão agrupados por pacote. A seguir é mostrado como os pacotes dependem entre si, e posteriormente o conteúdo de cada pacote.

5.2.1 Dependências entre os pacotes

O código-fonte foi dividido em 5 pacotes, web, service, utils, dao e model. O intuito da divisão foi agrupar as classes de acordo com suas funcionalidades, e separar as diferentes camadas do sistema. As relações de dependência entre os pacotes estão na figura 7.

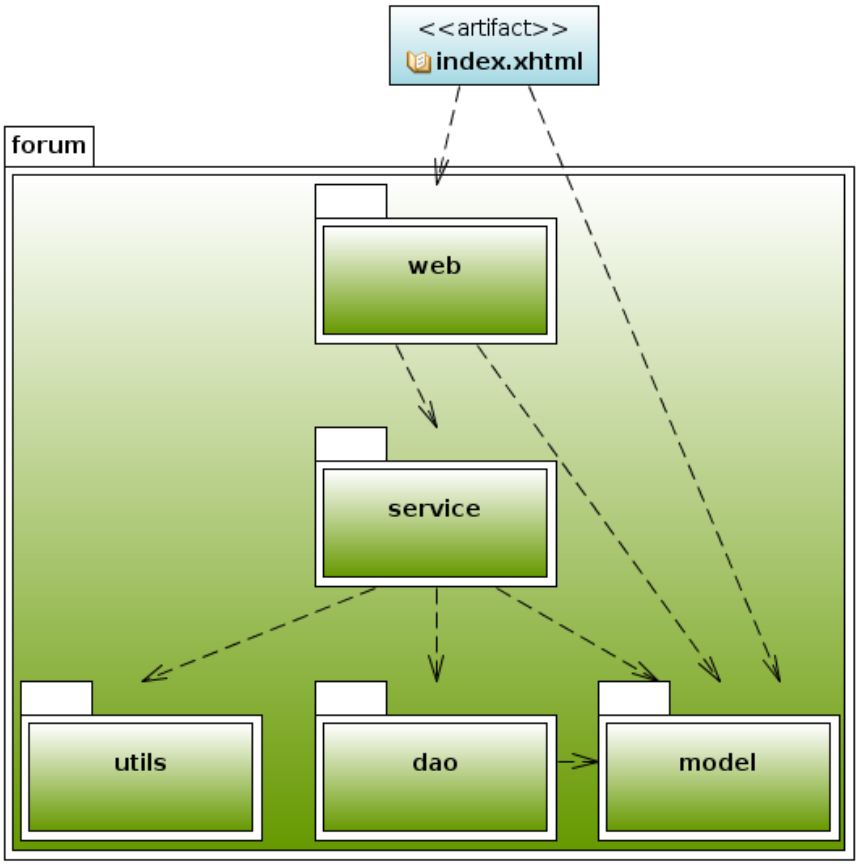

Figura 7. Dependências entre os pacotes do projeto.

5.2.2 Pacote Model

O pacote Model contém as entidades que compõem a ontologia de argumentação. O *schema* da base de dados é gerado automaticamente pela API JPA, com base nas classes deste pacote.

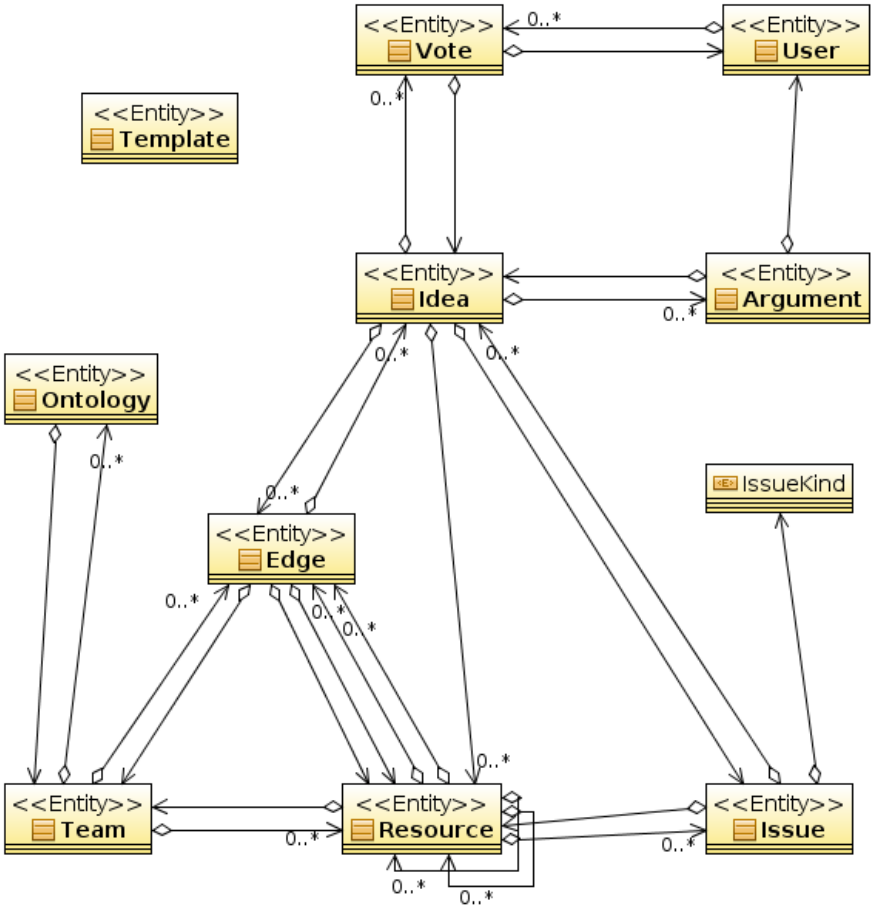

Figura 8. Diagrama de Classes do pacote Model*.*

5.2.3 Pacote DAO

O pacote DAO é composto por EJBs Stateless que utilizam o EntityManager da API JPA para consultar e persistir as entidades do pacote Model.

Existe um *facade* para cada entidade, cada *facade* realiza o *AbstractFacade*. As funções específicas destas classes geralmente executam consultas JPQL.

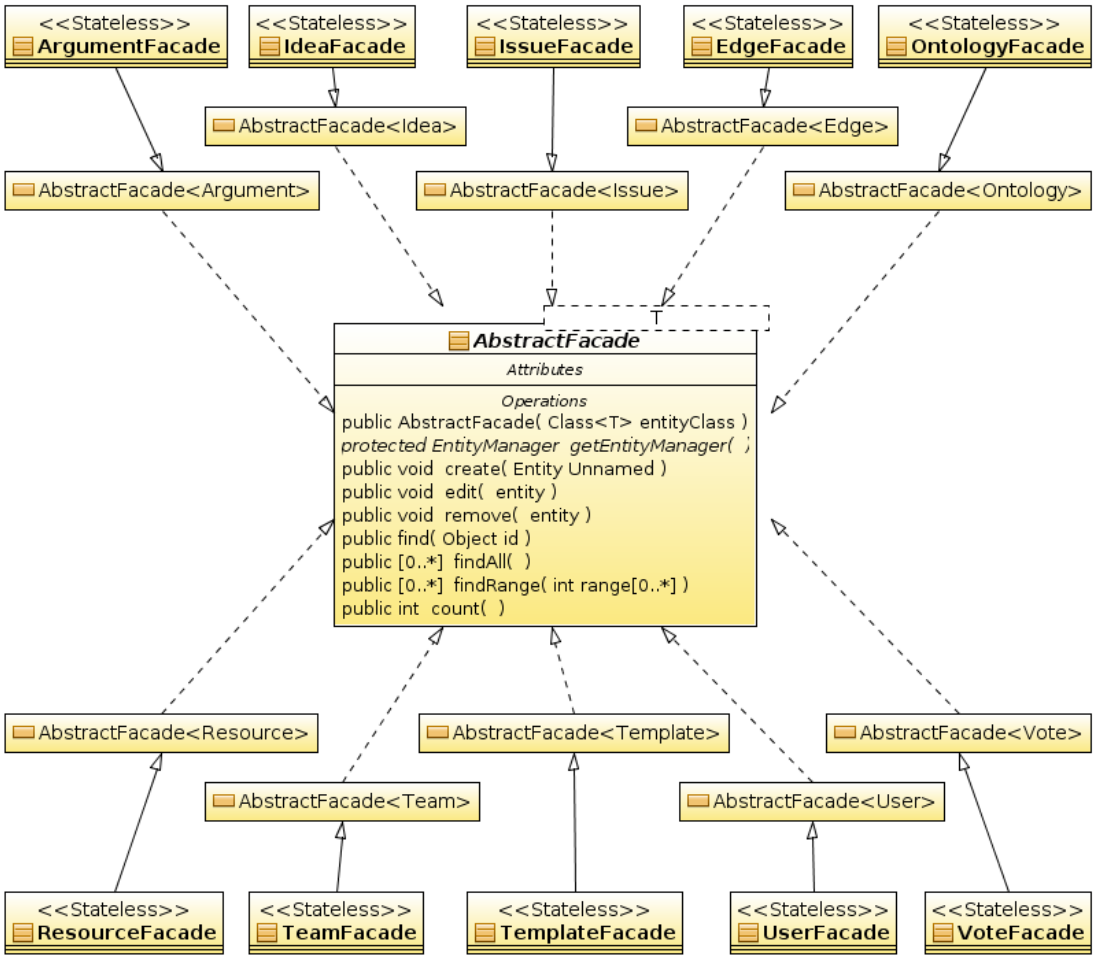

Figura 9. Diagrama de Classes do pacote DAO.

5.2.4 Pacote Service

As classes deste pacote equivalem a camada de controle no sistema, são utilizadas pelas classes do pacote web. Acessam as classes dos pacotes utils e dao. Existe um service para cada entidade. As classes são anotadas como EJBs Stateless.

5.2.5 Pacote Utils

Este pacote contém as classes OntologyUtils e OntoCleanUtils. Possuem funções específicas para auxiliar o processamento de ontologias, e avaliar as restrições impostas pela metodologia OntoClean. Dependem da OWL API.

5.2.6 Pacote Web

O pacote Web possui as classes que são a interface entre as páginas web e o sistema. Todas as funções públicas são acessíveis por páginas HTML e seus retornos podem ser manipulados através de tags EL (Expression Language). Assim como os pacotes service e dao, possui uma classe para cada entidade. As classes são anotadas como NamedBeans e SessionScoped.

5.2.7 Web Pages

No projeto existe uma pasta chamada Web Pages, o servidor de aplicação disponibiliza acesso ao conteúdo desta pasta através do protocolo HTTP.

As páginas Web (arquivos XHTML) formam a interface do usuário e permitem acesso ao sistema através dos NamedBeans do pacote web. A próxima figura mostra a inclusão das páginas e recursos a partir da página index.xhtml, sendo esta a única página acessada pelos usuários, as demais páginas são composições carregadas ou removidas em tempo de execução por meio de requisições AJAX (Asynchronous JavaScript and XML).

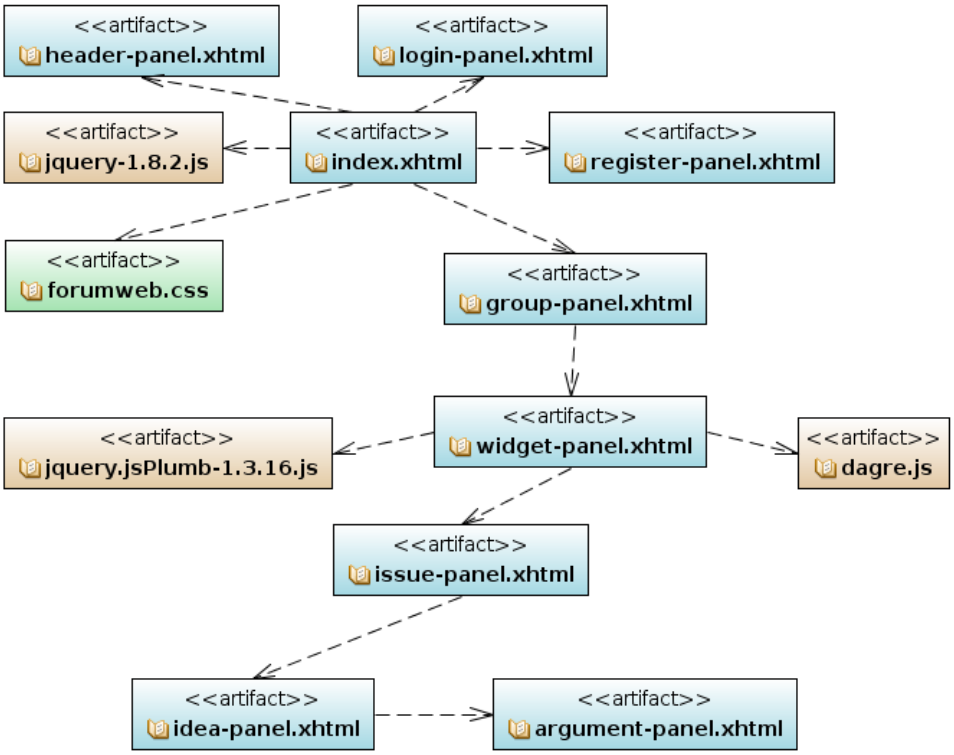

Figura 10. Arquivos XHTML, JS e CSS.

5.3 Protótipo

A seguir são apresentadas as telas do protótipo para os principais casos de uso. Os textos descrevem a sequência de eventos no sistema.

5.3.1 Login

O usuário entra com o nome e a senha e caso exista na base de dados, o objeto usuário é retornado e armazenado na sessão como o usuário conectado.

5.3.2 Cadastrar Grupo

Um grupo pode ser criado por qualquer usuário conectado no sistema. Os parâmetros são o nome e opcionalmente a descrição para o grupo. A tela pode ser vista na figura 11.

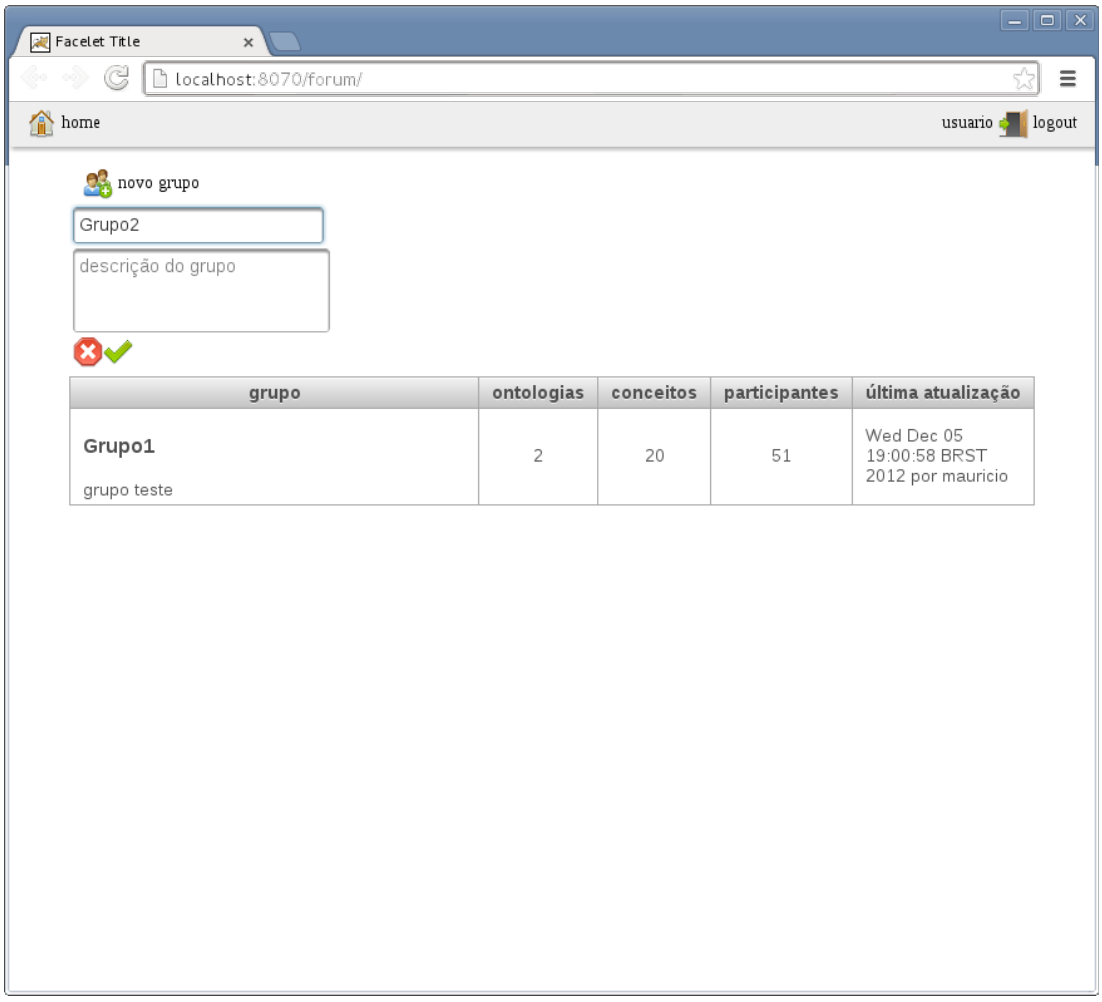

Figura 11. Tela Cadastrar e Listar Grupos.

5.3.3 Importar Ontologia

O processo é iniciado a partir de um grupo, os participantes precisam fazer o *upload* de arquivos OWL. A execução importa todos os elementos da ontologia carregada pelo usuário.

A esquerda da página do grupo (ver figura 12) podem ser vistos dois arquivos .owl importados (domínio arvore genealógica). A direita da página fica o diagrama com os conceitos importados. As setas representam as ideias taxonômicas. As setas tracejadas estão sendo desconsideradas, enquanto que, as contínuas estão sendo escolhidas (por número de votos).

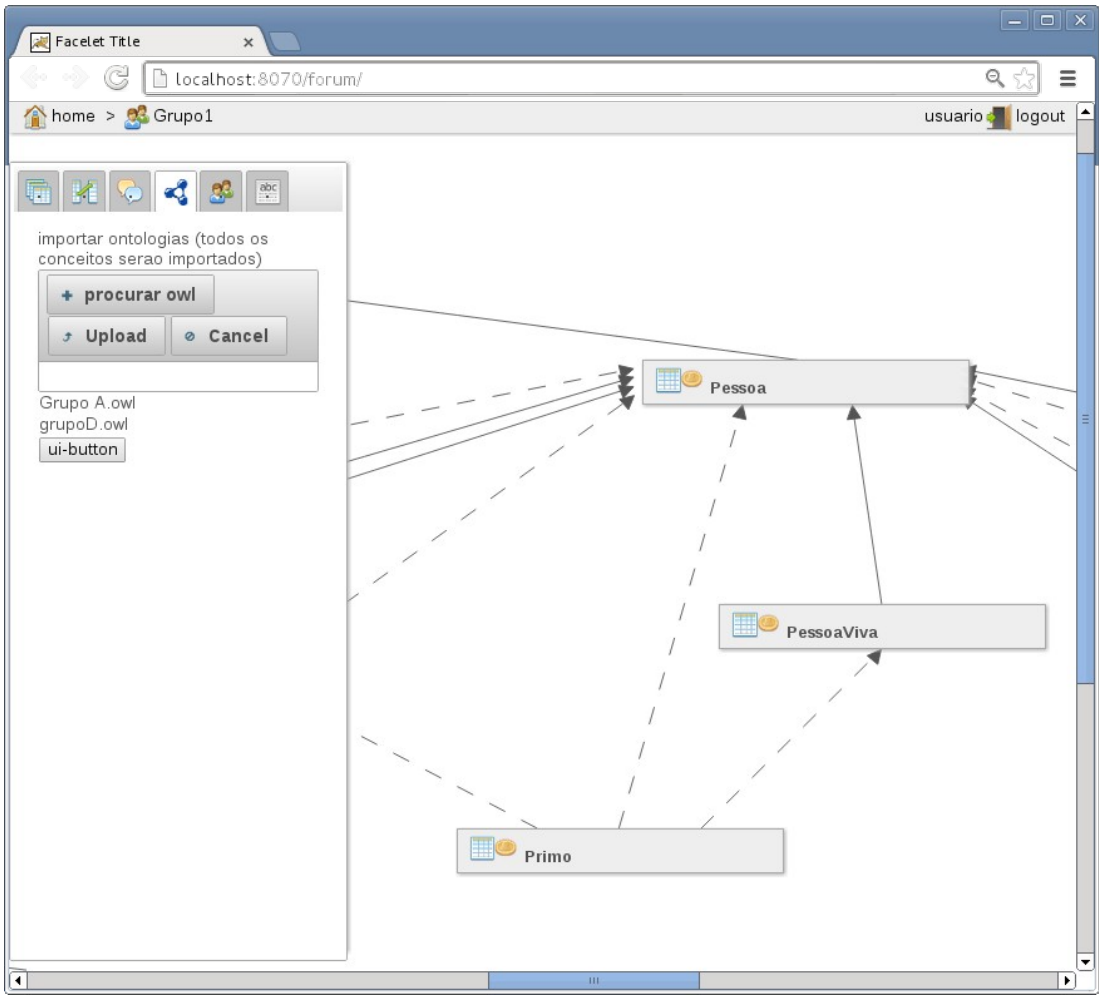

Figura 12. Tela Importar Ontologia.

5.3.4 Propor Ideia

Apenas ideias taxonômicas podem ser propostas, já que todas as ideias possíveis para as questões OntoClean são sempre criadas inicialmente. Para criar uma ideia o usuário precisa estar com um recurso e sua questão taxonômica aberta.

Por exemplo, na figura 13 é possível ver que a questão taxonômica do conceito Avô está aberta. Na questão há um menu de ítens com o conceito Pais pré-selecionado. É possível criar uma nova ideia (Avô subClassOf Pais) pressionando o ícone da lâmpada com o sinal +.

Na mesma figura ainda é possível ver que o conceito já possui duas ideias taxônomicas, a primeira (Avô subClassOf Pessoa), e a segunda (Avô subClassOf Antepassado).

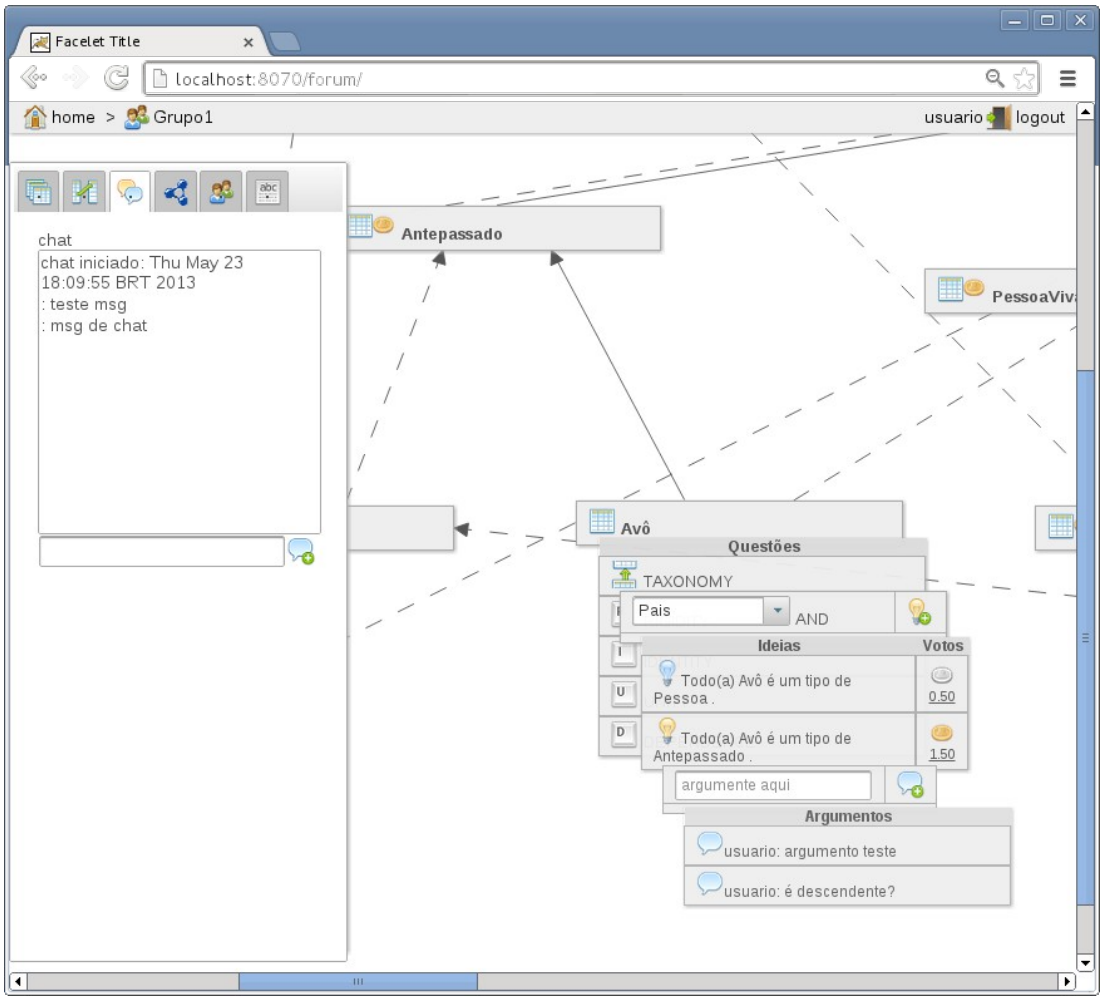

Figura 13. Tela Propor Ideia, Argumentar e Votar

5.3.5 Argumentar

Para argumentar um usuário deve clicar em uma ideia qualquer e então um campo de texto surge para que o usuário digite seu argumento. Os participantes do grupo também podem se comunicar através do chat. Os sistemas de argumentos e chat podem ser vistos na figura 13.

5.3.6 Votar

Para um usuário votar basta abrir uma questão e clicar no campo votar de alguma ideia. Após a contabilização de um voto, ocorre o processamento da votação (identificação da ideia vencedora) e também a validação das restrições OntoClean. O sistema de votação pode ser visto também na figura 13.

5.3.7 Validar Restrições OntoClean

A validação das restrições ocorre após cada voto computado, através da verificação de todas as ideias taxonômicas (setas) relacionadas ao recurso da questão recém votada.

Para cada relação taxonômica são verificadas algumas condições (ver tabela 11) nos valores das metapropriedades da origem (source) e do destino (target) da seta. A ideia representada por uma seta tem a forma (source subClassOf target). Quando uma condição é verdadeira, a seta é sinalizada com símbolos de aviso e explicações específicas à condição violada (ver figura 14).

Tabela 11. Condições de violação das restrições OntoClean

Condição
(source.getD().equals("-D") && target.getD().equals("+D"))
(source.getR().equals("+R") && (target.getR().equals("-R") || target.getR().equals("~R")))
(source.getR().equals("-R") && target.getR().equals("~R"))
(source.getI().equals("-I") && target.getI().equals("+I"))
(source.getU().equals("-U") && (target.getU().equals("+U") || target.getU().equals("~U")))
(source.getU().equals("+U") && target.getU().equals("~U"))
(source.getU().equals("~U") && target.getU().equals("+U"))

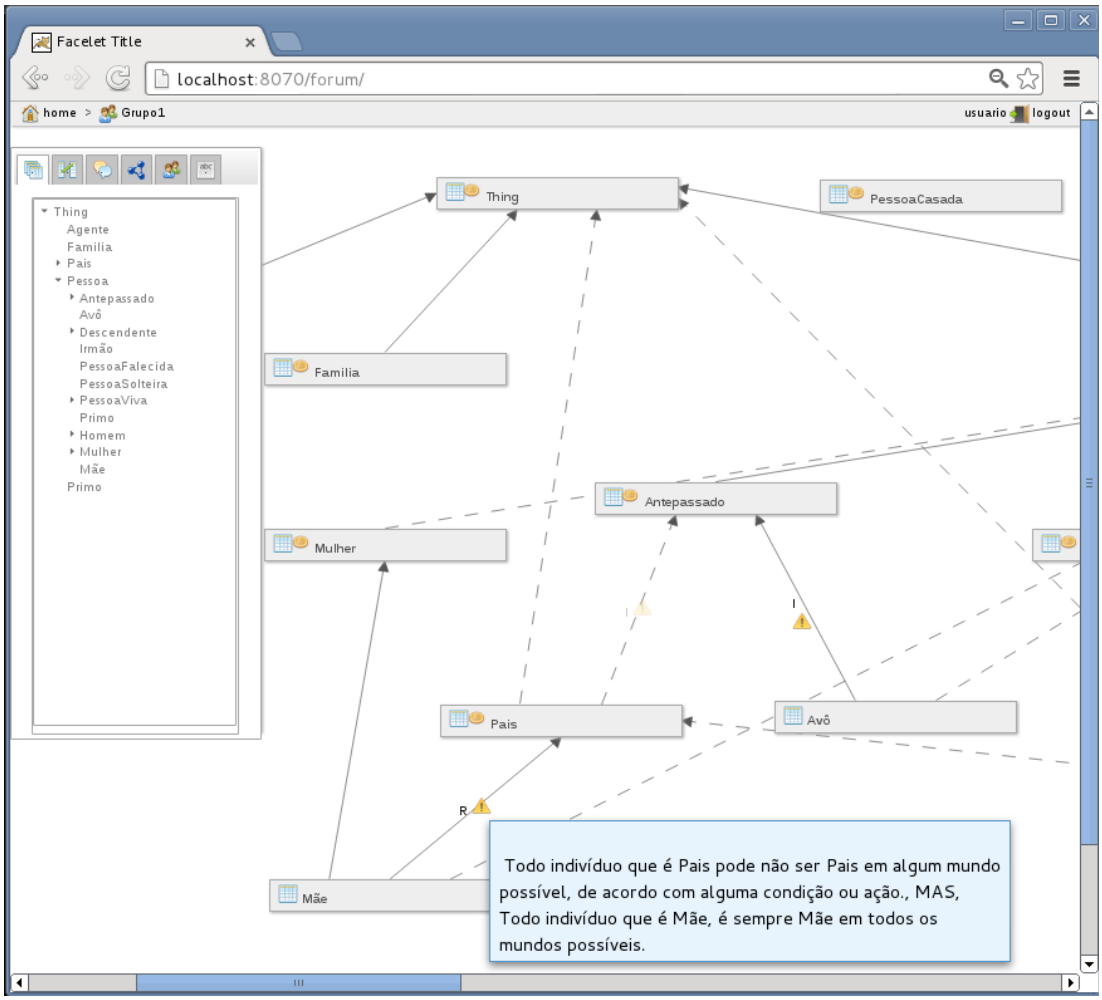

Figura 14. Tela Validar Restrições OntoClean.

6 Discussão

Os experimentos contribuíram muito durante o desenvolvimento, sendo que a maior parte das sugestões feitas pelos participantes, foram consideradas vindo a melhorar a experiência do usuário.

O tratamento de violações das restrições OntoClean mudou bastante no decorrer do projeto, inicialmente as questões envolvidas na violação tinham a votação anulada. Os participantes sugeriram que o sistema apenas sinalizasse as violações, mantendo os votos.

Embora o projeto tenha passado por várias modificações, o modelo para a Ontologia de Argumentação permaneceu praticamente o mesmo.

Seria interessante anotar a ontologia colaborativa exportada com comentários sobre o processo que ocorreu na ontologia de argumentação, uma espécie de histórico do desenvolvimento.

As metapropriedades OntoClean ainda não possuem um padrão para serem representadas. Para importar e exportar arquivos OWL, o projeto decidiu ler e gravar as metapropriedades utilizando labels para as classes. Isto não garante a compatibilidade entre diferentes sistemas que suportem a OntoClean.

A maior diversidade de situações acontece no tratamento da colaboração. É preciso controlar a sincronia das informações entre os participantes. Algumas situações podem levar a “falhas” decorrentes do tempo em que ocorreram as ações dos usuários. Por exemplo, ideias e argumentos criados após outros usuários já terem votado. O que parece ser mais prático é deixar o andamento da discussão totalmente livre, e utilizar recursos de notifcação como os *hot-topics* quando alguma alteração significativa ocorrer.

7 Conclusões

O melhor resultado obtido com este trabalho foi toda a experiência adquirida no decorrer do projeto.

O principal motivo para o desenvolvimento deste projeto foi a necessidade de uma ferramenta para auxiliar na avaliação do método CAMIO. O protótipo conseguiu cumprir os objetivos e foi utilizado como ferramenta. Os detalhes sobre a avaliação do método podem ser consultados no trabalho (DALL'AGNOL 2013).

Este motivo permitiu delimitar bastante o escopo de desenvolvimento. O protótipo foi desenvolvido considerando o fato de sempre rodar em um ambiente controlado nos experimentos. Isto possibilitou uma redução na quantidade de problemas a serem necessariamente tratados, mas que ocorreriam em um ambiente livre e público.

Posteriormente o trabalho pode ser aprimorado de modo a contemplar o desenvolvimento de ontologias de forma integral, adicionado suporte às relações, propriedades, restrições e instâncias.

A integração com fontes de dados como a LOD, abriria inúmeras oportunidades a serem exploradas. Desde a importação de conceitos até a aplicação das melhores práticas para publicar e conectar dados na Web.

8 Referências Bibliográficas

FREDDO, Ademir Roberto. Folkoncept: método de suporte à modelagem conceitual de ontologias a partir da aquisição de conhecimentos de folksonomias. 2010b. 225 f. Tese (Doutorado em Engenharia Elétrica e Informática Industrial) – Universidade Tecnológica Federal do Paraná, Curitiba, 2010b.

DALL'AGNOL, Josiane Michalak Hauagge. Um Método de suporte à argumentação bem-fundamentada para modelagem conceitual no desenvolvimento colaborativo de ontologias. 2011. 113 f. Texto de Qualificação – Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI), Universidade Tecnológica Federal do Paraná (UTFPR). Curitiba, 2011. (A ser publicado)

ODM. Ontology Definition Metamodel Version 1.0. OMG Document Number: formal/2009-05-01 Standard document URL: http://www.omg.org/spec/ODM/1.0 (acesso em 15/05/2013).

GUARINO, Nicola. WELTY, A. Christopher. An Overview of OntoClean. 2004.

WELTY, A Christopher. OntOWLClean: Cleaning OWL ontologies with OWL. 2006.

GUARINO, Nicola. Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, pp. 3-15.

BAADER, Franz. NUTT, Werner. Basic Description Logics. The Description Logic Handbook - Theory, Implementation and Applications 2003, pp. 47-100.

BIZER, C., HEATH, T., BERNERS-LEE, T. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5, (3), pp. 1-22. 2009.

BOLLACKER, K., TUFTS, P., PIERCE, T., COOK, R. A Platform for Scalable, Collaborative, Structured Information Integration. 2007.

WIKIPEDIA. http://en.wikipedia.org/wiki/Wikipedia (acesso em 15/05/2013)

DBPEDIA. http://dbpedia.org/About (acesso em 15/05/2013)