UNIVERSIDADE DO ESTADO DA BAHIA

DEPARTAMENTO DE CIÊNCIAS EXATAS E DA TERRA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANDRÉ ARAÚJO SILVA

Especificação e Construção do Editor Gráfico de Ontologias para Automatizar o MOBI

Salvador - BA - Brasil

2013

ANDRÉ ARAÚJO SILVA

Especificação e Construção do Editor Gráfico de Ontologias para Automatizar o MOBI

Trabalho de Conclusão de Curso apresentado a Universidade do Estado da Bahia-UNEB, Departamento de Ciências Exatas e da Terra, Campus I, como pré-requisito para a conclusão do curso de Sistemas de Informação.

Orientador: Prof. Dr. Eduardo Manuel de Freitas Jorge.

Salvador - BA - Brasil

2013

Resumo

O modelo computacional baseado em Ontologia é um dos mais recentes na área da Ciência da Computação. Entretanto, diversas pesquisas, métodos de modelagem e ferramentas foram e continuam sendo criados para este modelo. Dentre estas ferramentas, uma das mais recentes e inovadoras é a MOBI (Método de Modelagem de Ontologia Baseado em Instância), que propõe uma mudança paradigmática na modelagem baseada em Ontologia. Diante da necessidade do conhecimento da linguagem de programação Java, observou-se a necessidade da especificação / implementação de uma ferramenta que automatizasse o processo de modelagem tornando menor o nível de complexidade para a modelagem de Ontologias usando o método citado. Baseado nisso o objetivo desse projeto é especificar, construir e validar um protótipo de um editor de ontologias baseado no MOBI chamado Editor MOBI.

Palavras-chaves: MOBI, Ontologia, Modelagem, Editor Gráfico.

Abstract

The computational model based on Ontology is one of the latest in the field of Computer Science. However, several studies, modeling methods and tools have been and continue to be created for this model. Among these tools, one of the newest and innovative is the MOBI (Modeling Method Based on Ontology Instance), who proposes a paradigm shift in Ontology-based modeling. Given the need for knowledge of the Java programming language, there is a need of specification / implementation of a tool that automates the modeling process becoming the lowest level of complexity for modeling ontologies using the method mentioned. Based on this the goal of this project is to specify, design and validate a prototype of an ontology editor based on the method MOBI, called Editor MOBI.

Key-words: MOBI, Ontology, Modeling, Graphical Editor.

Lista de ilustrações

Figura 1 – Etapa que o MOBI está relacionada com o método proposto por Uschold. 18
Figura 2 – Macro Fluxo do Processo Convencional de Codificação de Ontologia. (adaptado de JORGE, 2010) 19
Figura 3 – Processo de Codificação de Ontologia Baseado em Instâncias. (adaptado de JORGE, 2010) 19
Figura 4 – Arquitetura da construção do protótipo do editor gráfico. 21
Figura 5 – Protótipo de tela. 23
Figura 6 – Itens da interface gráfica numerados. 24
Figura 7 – Instâncias adicionadas no quadrante esquerdo. 25
Figura 8 – Adicionadas as instâncias nos quadrantes da direita e da esquerda. . . . 26
Figura 9 – Relacionamentos criados. 26
Figura 10 – Adicionando os nomes das classes. 27
Figura 11 – Adicionando os nomes da relação. 27
Figura 12 – Devolução do tipo de relacionamento. 28
Figura 13 – Exibição da cardinalidade da relação. 28
Figura 14 – Tela do editor Gráfico. 29
Figura 15 – Instâncias Adicionadas 30
Figura 16 – Relacionamentos adicionados a modelagem 30
Figura 17 – Relacionamento recém definido na interface gráfica 31
Figura 18 – Classes definidas 31
Figura 19 – Retorno do tipo de relação 36
Figura 20 – Cardinalidade e nomes das relações definidos. 37
Figura 21 – Arquivo OWL produto da modelagem 38
Figura 22 – Modelagem da relação entre Candidato e Partido 42
Figura 23 – Identificação do arquivo da primeira modelagem no Protége 42
Figura 24 – Identificação do arquivo da segunda modelagem no Protége 43
Figura 25 – Classes no arquivo Validacao2Generico2.owl 43
Figura 26 – Classes no arquivo Generico1.owl 44
Figura 27 – Validacao2Generico2.owl propriedade funcional 44
Figura 28 – Validacao2Generico2.owl propriedade inversa funcional 45
Figura 29 – Generico1.owl propriedade funcional 45
Figura 30 – Generico1.owl propriedade inversa funcional 46
Figura 31 – Intâncias da classe "Candidato"do arquivo Validacao2Generico2.owl. . . 46
Figura 32 – Intâncias da classe "Partido"do arquivo Validacao2Generico2.owl. . . . 47
Figura 33 – Intâncias da classe "Candidato"do arquivo Generico1.owl. 47

Figura 34 – Intâncias da classe "Partido"do arquivo Generico1.owl. 48

Lista de abreviaturas e siglas

API	Application Programming Interface.
DAML	DARPA Agent Markup Language.
GUI	Interface Gráfica do Usuário.
IDL	Interface Description Language.
LAL	Léxico Ampliado da Linguagem.
MOBI	Modelo de Ontologia Baseado em Instância.
OIL	Ontology Inference Layer.
OWL	Web Ontology Language.
SHOE	Simple HTML Ontology Extensions.
XOL	Ontology Exchang Language.

Sumário

Contextualizando o Projeto **9**

1 O Que é Necessário Saber **12**

1.1 Ontologia – O Estudo do Conhecimento 12

1.2 Composição e classificação das Ontologias 14

1.3 O produto final do Editor MOBI 15

1.4 OWL – Web Ontology Language 15

1.5 A Necessidade de um Método para Modelagem 17

1.6 MOBI – Modelagem de Ontologias Baseado em Instâncias 17

1.7 Sobre o MOBI 17

2 O Projeto do Editor de Ontologias MOBI **21**

2.1 Projeto arquitetural de alto nível e uma pequena descrição 21

2.2 Forma de trabalho 22

2.3 Protótipo Funcional 23

2.4 Ferramentas de Desenvolvimento 28

2.5 Editor MOBI na Prática e Detalhes do Desenvolvimento 29

2.6 Limitação do Projeto 38

3 Validação do Projeto **39**

4 Considerações Finais e Recomendações **49**

4.1 Conclusão 49

4.1.1 Perspectivas Futuras 50

Referências **51**

Contextualizando o Projeto

Na contemporaneidade as informações são disseminadas com grande velocidade em decorrência do alto nível de conectividade experimentado pelas pessoas, o que viabiliza um amplo acesso a um grande volume de informações de forma rápida e quase simultânea. Uma parte considerável da população passa muito tempo conectada ao mundo virtual mediante uso de computadores, tablets e aparelhos celulares via internet móvel ou redes Wi-Fi. Segundo o (FOLHA. . . , 2013),no Brasil o acesso à internet cresceu 143,8% no período de 2005 a 2011 entre a população com 10 anos ou mais.

Nesse mundo extremamente conectado as informações muitas vezes são compartilhadas sem qualquer filtro ou classificação lógica. Numa pesquisa num site de busca, por exemplo, um conceito simples pode estar associado há milhões resultados. Isso faz com que, encontrar de maneira assertiva o que de fato se esta buscando, seja um processo muito mais custoso. Em decorrência disso, o aperfeiçoamento dos sistemas computacionais voltados para as técnicas de organização da informação para o gerenciamento desse grande volume de informação que pode estar, por exemplo, na web ou nos banco de dados das empresas, tem ganhado uma importância cada vez maior, devido ao alto nível de complexidade para sua sistematização e manutenção.

Pensando sob a óptica da organização da informação, a fim de viabilizar acesso e possível compreensão, a questão central é: as pessoas representam os seus conceitos de maneiras completamente diferentes, sendo que a forma dessa representação irá depender única e exclusivamente da forma de raciocinar do indivíduo que esta construindo a representação. A partir das variadas interpretações e raciocínios dos indivíduos, um mesmo conceito pode representar diferentes conteúdos, dessa forma, existem então inúmeras possibilidades de representação das informações das mais diversas temáticas, associando-se ao fato de que não há o uso da semântica para guiar a busca por essas informações, surge à necessidade de desenvolvimento de um mecanismo formal que possibilite à construção de estruturas computacionais aplicadas a recuperação da informação contida nas bases de conhecimento.

Quando se pensa em recuperação da informação, deve-se levar em consideração o tema ontologias, devido sua capacidade de sistematizar informações estabelecendo relações entre termos e conteúdos associados a eles. O uso de ontologias vem se destacando no campo das pesquisas para a organização de informações sobre determinado domínio de conhecimento, isso devido a sua potencialidade em organizar e representar a informação. Uma ontologia “define as regras que regulam a combinação entre termos e relações em um domínio do

conhecimento" (FORTE; SOUZA; PRADO, 2003, p. 3).

A construção de uma ontologia, no entanto, não é algo tão simples, uma vez que além de ser necessário conhecer o domínio a ser modelado, é de fundamental importância também conhecer os fundamentos das linguagens formais, metodologias e ferramentas de modelagem. Tudo isso torna o processo bastante custoso, sobretudo devido à necessidade do conhecimento do domínio a ser modelado, o que nos remete a um caráter artesanal ao invés de um processo com etapas definidas e claras que poderiam ser descritas assim como um processo científico.

Existem métodos para a modelagem de Ontologia, por exemplo: LAL, Kactus, Methontology, Método 101, Cyc, Uschold e outros. Eles possuem diferenças entre si, contudo um princípio é comum a todos, a codificação da estrutura da Ontologia começando através das classes, seus relacionamentos e axiomas para posteriormente identificar os objetos (instâncias) (BREITMAN, 2005).

A partir de inquietações referentes aos métodos citados anteriormente, Jorge (2012) propôs o método denominado MOBI (Modelagem de Ontologia Baseado em Instância), método este, que propõe à alternativa de modelar cenários, a partir das relações entre instâncias, daí então chegando até as relações entre classes (Buttom-up). Dentro do panorama da pesquisa já existente em relação ao método MOBI, ainda existe a necessidade de um editor desktop a fim de automatizar o processo de modelagem, tornando o nível de complexidade para a modelagem de ontologias menor, por não exigir mais a necessidade do conhecimento da linguagem de programação compreendida pelo núcleo do MOBI.

O objetivo dessa pesquisa é especificar, construir e validar um protótipo de um editor de ontologias baseado no MOBI. Durante o trabalho adotaremos "Editor MOBI"como nome do editor de Ontologias baseado no método MOBI que esta sendo especificado. Para a execução do trabalho primeiramente será feito uma revisão de literatura sobre ontologias, assunto de fundamental importância para o domínio e apropriação do tema da pesquisa, sendo seguido por uma apresentação de conteúdos relevantes sobre OWL, pois foi adotado como produto final da modelagem de uma Ontologia no Editor MOBI que esta sendo especificado um arquivo OWL, já que é um dos possíveis produtos gerados pela ferramenta de modelagem do MOBI (kernel MOBI), já existente e de funcionamento comprovado. Esse OWL resultante da modelagem será utilizado nesse trabalho para fins de avaliação da eficiência do processo do método como um todo. Além disso, será feita também uma apresentação sobre os conceitos do método MOBI, visando demonstrar os conteúdos que fazem parte do universo da modelagem de Ontologias e que são de fundamental importância para o entendimento do funcionamento do Editor MOBI.

Após a demonstração teórica dos assuntos relevantes ao objeto da pesquisa, então será

apresentado o trabalho de especificação e implementação do "Editor MOBI", que será implementado utilizando a linguagem de programação Java, maiores explicações sobre a construção do editor serão oferecidos nos itens referentes ao desenvolvimento do projeto.

Para a validação do projeto será modelado um universo de discurso do domínio de uma “Eleição”, usando o "Editor MOBI"especificado neste trabalho. Essa mesma modelagem também será realizada de maneira manual, na linguagem de programação entendida kernel do MOBI. Como resultado serão obtidos dois arquivos OWL de descrição das Ontologias modeladas. Estes dois arquivos serão importados para o software Protégé, que é uma plataforma que dá suporte a modelagem de Ontologias, cada um por sua vez, então poderão ser comparadas as estruturas das duas Ontologias criadas, através da observação dos dados apresentados no Protégé. Dessa forma o Editor MOBI poderá ser avaliado quanto a sua assertividade na modelagem da Ontologia.

Espera-se ao final do trabalho ter um protótipo de um editor voltado para o método MOBI, devidamente especificado e implementado, atendendo as necessidades do modelo e funcionando de forma satisfatória.

1 O Que é Necessário Saber

A partir daqui serão abordados de maneira rápida e direta os temas que tem envolvimento com a modelagem de ontologias realizada no Editor MOBI, a fim de prover o conteúdo necessário acerca dos temas que circundam este trabalho para o leitor. Sobre a óptica da organização da informação uma das teorias que tem ganhado mais espaço é a Ontologia. Este trabalho tem por base a organização da informação através da modelagem de Ontologias, obtendo como produto final da modelagem, uma Ontologia descrita em um arquivo OWL, gerado com a aplicação do método MOBI, logo, torna-se necessário que o leitor conheça um pouco sobre os temas Ontologia, OWL e MOBI.

1.1 Ontologia – O Estudo do Conhecimento

O uso do termo Ontologia foi utilizado no século XVII pelo filósofo alemão Jacobus Thomasius, no entanto sua origem etimológica tem na sua composição uma palavra, derivada de dois substantivos gregos (Onto: Ser e Logos: conhecimento, estudo) que na sua essência trazem o significado da busca pelo entendimento do ser, tal como ele é e não como parece ser. Sob influências de uma perspectiva filosófica a palavra Ontologia adquire o significado de o estudo do Ser, das coisas ou dos entes, tais como são em si mesmas, real e verdadeiramente (OYOLA; ALVARENGA, 2009).

Para Christian Wolff (1679-1754) e Alexander Gottlieb Baumgarten (1714-1762), a ontologia trata do ser enquanto ser. Dentro do contexto da Ontologia a filosofia tenta responder a questões como: “O que é um ser?”, ou ainda, “Quais as características comuns a todos os seres?” (MAEDCHE, 2002 apud GUIMARãES, 2002).

A Ontologia e o tratamento das informações situam-se entre a ciência da informação e a ciência da cognição, recebendo também influência de outras áreas como: biblioteconomia, comunicação, e ciência da computação. Cabendo a ciência da informação a função de procurar desenvolver metodologias que permitam a elaboração de ontologias aplicadas aos diferentes campos do saber.

Caminhando para o universo da Computação, por Ontologia entende-se uma capacidade de mapear o conhecimento de uma área específica, categorizar, organizar ou recuperar informações quando requisitado. Segundo Silva, Souza e Almeida (2010, p. 2):

> As ontologias apresentam-se como possibilidades de representação de conhecimento em sistemas de informação na medida em que buscam

> organizar e padronizar conceitos, termos e definições aceitas por uma comunidade particular.

O conceito de Ontologia foi incorporado à ciência da computação com o objetivo de estudar formas para organização e recuperação da informação, sendo reconhecida como uma grande área de pesquisa para a representação e organização do conhecimento. Descrevendo ontologia, Almeida (2006, p. 106) afirma que:

> O estudo de Ontologias caracteriza-se como um ramo de pesquisa que surgiu no final dos anos 80, propondo alternativas para representar os conceitos, as relações entre os conceitos e a semântica de um domínio do conhecimento. A semântica, nesse contexto, é a parte de um modelo formal em que declarações lógicas representam o conhecimento do domínio a ser manipulado em um sistema computacional.

Assim sendo, uma ontologia tem a intenção de descrever certo domínio do conhecimento, especificando-o a fim de torná-lo compreensível para humanos e sistemas computacionais. Dentro dessa visão de ontologia (GUARINO, 1998 apud ALMEIDA, 2006) afirma que :

> [...] Uma ontologia refere-se a um artefato de engenharia (de software) que é constituído por um vocabulário específico utilizado para descrever certa realidade, mais um conjunto de suposições explicitas a respeito do significado pretendido para as palavras do vocabulário[...]

A definição de Guarino ajuda na compreensão do objetivo de uma Ontologia na prática, inserida no ambiente da ciência da computação, enfatizando a existência de um vocabulário dentro de uma Ontologia que define um determinado domínio do conhecimento.

Com o aumento do número de documentos e arquivos variados nas bases de conhecimento, podendo tomar como exemplo a grande base de conhecimento internet, surge a necessidade de uma organização mais precisa para a obtenção mais assertiva no momento da recuperação da informação. Para obter essa organização podem ser utilizadas as Ontologias, que hoje já são utilizadas em projetos nas diversas áreas do conhecimento como, por exemplo, gestão do conhecimento, comércio eletrônico, processamento de linguagens naturais, recuperação de informação na web, entre outros.

Capturar as informações geradas pelo conhecimento humano e torná-las explícitas em Ontologias não é algo trivial. Daí a importância de um processo de sistematização e recuperação de informações para qualquer área do conhecimento. Para a construção de um arquivo que define uma Ontologia utilizam-se como recursos, linguagens específicas para

essa finalidade. Para o desenvolvimento desse trabalho, como já foi mensionado será adotada a Web Ontology Language - OWL, visto que o kernel já existente do MOBI gera um arquivo OWL. Maiores detalhes sobre essa linguagem seram explanados posteriormente.

1.2 Composição e classificação das Ontologias

Ontologias normalmente têm a mesma estrutura, sendo compostas por:

- Classes - são entidades que representam determinado conceito e são organizadas hierarquicamente;
- Relações – representam a interação entre os conceitos;
- Instâncias - representam os objetos pertencentes a classes;
- Axiomas – são representações formais, que possibilitam a manipulação por sistemas computacionais a partir de considerações sobre sua semântica

Os axiomas são as sentença iniciais consideradas verdadeiras. Para Silva (2008, p. 142-143): “São definidas como sentenças em primeira ordem usando predicados da ontologia”. Um exemplo de axioma seria:

O ônibus local é uma sub-classe de ônibus
∀ $x(ônibus_ local($x) -> ônibus ($x));

Ontologias podem ser classificadas de diversas formas, tomando por base vários aspectos, por exemplo: Heijst, Schreiber e Wielinga (1997) classificam as ontologias, quanto ao tipo de estrutura e ao assunto da conceitualização; Haav e Lubi (2001) classificam as ontologias, quanto aos tipos de classes; Uschold e Jasper (1992) classificam as ontologias de acordo com sua função no processo de desenvolvimento de sistemas computacionais; e Uschold e Gruninger (1996) classificam a ontologia de acordo com o grau de formalidade utilizado para especificar o vocabulário de termos e seus significados. Obedecendo a seguinte escala de classificação:

- Ontologia altamente informal - em que o vocabulário é expresso em linguagem natural;
- Ontologia semi-informal - em que o vocabulário é expresso em linguagem natural de forma restrita e estruturada;

- Ontologia semi-formal - cujo vocabulário é expresso em linguagem artificial definida formalmente;
- Ontologia rigorosamente formal - em que os termos são definidos com semântica formal, teoremas e provas.

1.3 O produto final do Editor MOBI

Uma vez modelada a Ontologia no editor MOBI será obtido como produto da modelagem de um domínio um arquivo OWL, que descreve a Ontologia modelada. Por isso o leitor irá conhecer agora um pouco sobre os conceitos do OWL.

1.4 OWL – Web Ontology Language

Um projeto desenvolvido com base em ontologias visa melhorar a busca pelas informações, aumentando sua exatidão, possibilitando a recuperação dos dados relevantes para o usuário. No que se refere ao uso de ontologias para web é necessário o uso de uma linguagem compatível. Para o seu desenvolvimento é relevante o uso de linguagens que suportem estruturas para representação do conhecimento, isto é, que sejam constituídas de um agrupamento descritivo de termos referentes a um domínio específico. Lima e Carvalho (2005, p. 3) afirmam que: "As linguagens de ontologia para Web são, geralmente, expressas em uma linguagem lógica, a lógica descritiva, garantindo as distinções entre as classes, propriedades e relações, evitando ambiguidades".

Existem variadas possibilidades de linguagens para a descrição de Ontologias, dentre elas pode-se citar: SHOE (Simple HTML Ontology Extensions), XOL (Ontology Exchang Language), OIL (Ontology Inference Layer) e DAML (DARPA Agent Markup Language) e a OWL (Web Ontology Language).

Diante das possibilidades de linguagens a World Wide Web Consortium-W3C estabeleceu parâmetros para organização e padronização de linguagens a serem usadas, A partir de 2004 foi recomendado o uso da linguagem OWL. A linguagem OWL surge da união de duas linguagens: a européia, OIL (Ontology Inference Layer) e norte americana DAML (DARPA Agent Markup Language).

A Web Ontology Language (OWL) é uma linguagem que tem como característica a marcação semântica nas publicações e o compartilhamento de ontologias na web, desenvolvidas, sobretudo objetivando descrever e conceituar classes e os possíveis relacionamentos entre elas.

> Uma ontologia OWL pode formalizar um domínio, definindo classes e propriedades destas classes, definir indivíduos e afirmações sobre eles e, usando-se a semântica formal OWL, especificar como derivar conseqüências lógicas, isto é, fatos que não estão presentes na ontologia, mas são vinculados pela semântica (LIMA; CARVALHO, 2005, p. 4).

A OWL, entre suas principais características identifica-se uma facilidade para expressar aspectos semânticos, por não se restringir apenas há visualização, mas também o processamento da informação. Além disso, apresenta um perfil descritivo e lógico, com semântica definida evitando o uso de ambigüidades. (FORTE; SOUZA; PRADO, 2006a).

Para o desenvolvimento de uma ontologia em OWL, é importante o uso de elementos básicos, sendo eles: as classes, os relacionamentos, e instâncias. A OWL também aponta restrições às propriedades, sendo estas de valor ou de cardinalidade. Lima e Carvalho (2005) descrevem os elementos relevantes numa linguagem OWL:

1. CLASSES: agrupamento devido à características similares
    a) EXTENSÃO DE CLASSE: grupo de indivíduos associados a uma determinada classe
    b) INSTÂNCIAS: os indivíduos em uma extensão
2. PROPRIEDADE: relacionamento entre indivíduos, associando a fatos específicos. Características das propriedades: Transitivas, Simétricas, Funcionais, Inversas.

A linguagem OWL possui três tipos de sublinguagens, que são versões específicas, usadas de acordo a necessidade, são elas: a OWL Lite, OWL DL e OWL Full. A OWL Lite apresenta para os usuários uma hierarquia de classificação e de características de restrição simples. A OWL DL é indicada para usuários que necessitam do máximo de expressividade sem perder a garantia computacional, inclui todas as construções de OWL com restrições como separação de tipos, por exemplo: uma classe não pode ser também um indivíduo ou propriedade, uma propriedade não pode ser também um indivíduo ou classe. O DL de OWL DL é proveniente da correspondência com Lógica Descritiva. Vale ressaltar que para uma OWL DL com owl:DatatypeProperty não pode ser marcada como uma owl:InverseFunctionalProperty. Uma outra sub-linguagem é a OWL Full indicada para usuários que necessitam do máximo de expressividade, porém sem garantias computacionais. Uma OWL Full permite aumentar o significado do vocabulário pré-definido. É improvável que algum software de raciocínio seja capaz de implementar todas as funções da OWL Full.

1.5 A Necessidade de um Método para Modelagem

Para existir uma Ontologia de um determinado domínio é necessário que esta seja descrita em alguma das linguagens que proporcionam suporte para a descrição de ontologias. Para a realização dessa descrição existem os métodos para a escrita do arquivo de descrção da ontologia, um desses métodos é o MOBI que será apresentado a seguir.

1.6 MOBI – Modelagem de Ontologias Baseado em Instâncias

Conforme já foi comentado na introdução do trabalho, estruturar o conhecimento não é algo trivial, nesse contexto surge o desafio de como estruturar os conceitos, visando o armazenamento e a disseminação do conhecimento. As Ontologias são uma boa opção para alcançar esse objetivo, devido à capacidade de mapear o conhecimento. Apesar do surgimento recente das Ontologias como possibilidade de representação do conhecimento, na década de 90, o assunto já existe num nível de aprofundamento e disseminação dentro do cenário da computação.

Existem diferentes métodos e ferramentas para utilização e suporte dos conceitos de Ontologia, que se baseiam no paradigma de codificação baseado em classes. Assim o MOBI foi proposto por (JORGE, 2012) com o intuito tornar mais didático o aprendizado de como modelar um determinado domínio.

1.7 Sobre o MOBI

O MOBI (Modelagem de Ontologia Baseada em Instâncias) é um método que propõe a mudança de paradigma de construção de Ontologias baseada em classes, para o paradigma baseado em instâncias.

Segundo o modelo proposto por Jorge (2012, p. 39):

> A proposta do MOBI é inverter a forma convencional de modelagem baseado em classes, identificando primeiramente um subconjunto de relações entre instâncias e assim através desse subconjunto determinar de forma automática as relações e algumas regras das futuras classes.

Para tanto existe uma premissa necessária para a existência do método, que diz que, toda instância é única e que suas classificações podem mudar a depender do domínio ou de alterações feitas por quem esta modelando. Assim, o elo semântico passar a ser a ligação entre as instâncias.

O MOBI busca reduzir o esforço no momento da transformação de uma abstração em notação formal, por conta da multidisciplinaridade existente nesse processo, que o torna uma atividade complexa.

Todas as demais metodologias possuem uma etapa de codificação que é comum a todas elas. Uma possível forma de dividir as etapas de construção de uma ontologia é proposta por Uschold e King (1995) que dividem essas etapas em três:

- Primeira – Identificar propósito, determinar o domínio de modelagem;
- Segunda – captura de conceitos para serem codificados e integrados em uma ontologia;
- Terceira – validação da ontologia e documentação;

Maiores detalhes sobre o método e a divisão podem sem encontrados no artigo de Uschold e King, "Towards a Methodology For Building Ontologies" (1995).

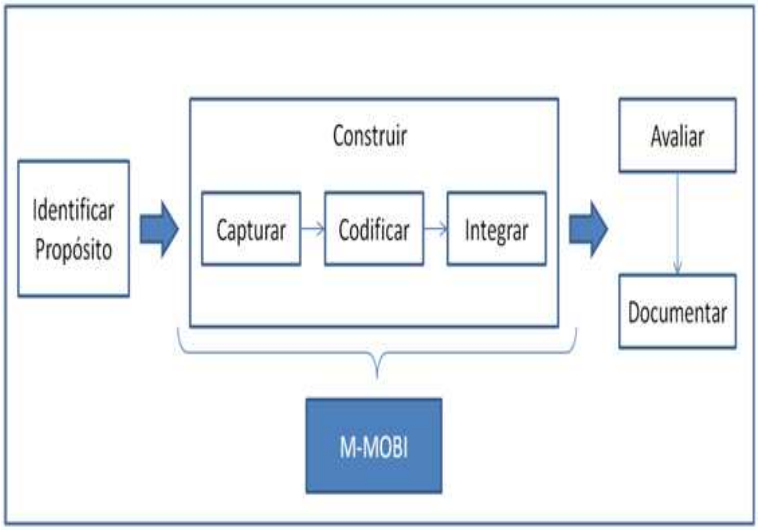

Figura 1 – Etapa que o MOBI está relacionada com o método proposto por Uschold.

Fonte: adaptado de (BREITMAN, 2005 apud JORGE, 2012)

O MOBI concentra-se na segunda etapa do método proposto por Uschold e King atuando na construção da Ontologia, mais precisamente no momento da codificação, como pode ser visto na figura 1. Nesta etapa, no momento da codificação, o processo convencional citado divide o processo de construção de Ontologias em três etapas principais, fluxo que é comum para todos os métodos de modelagem de Ontologia. Essas etapas são:

- Etapa 1: Definição das classes, suas hierarquias e axiomas;

- Etapa 2: Vinculação de instâncias (objetos);
- Etapa 3: Processo finalizado e Ontologia construída.

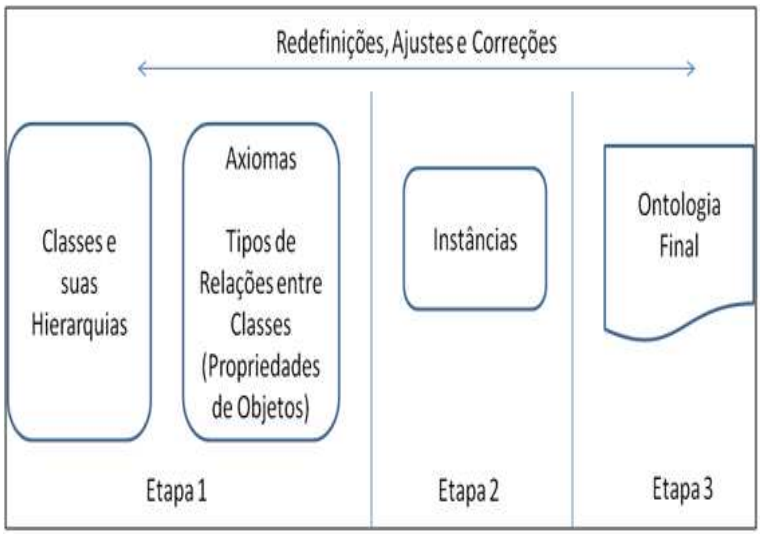

Figura 2 – Macro Fluxo do Processo Convencional de Codificação de Ontologia. (adaptado de JORGE, 2010)

Estando o MOBI na etapa 1, acontece a modelagem do conjunto de instâncias e suas relações criadas e classificadas; na etapa 2 o MOBI codifica automaticamente as regras que mapeiam as relações estruturais da Ontologia.

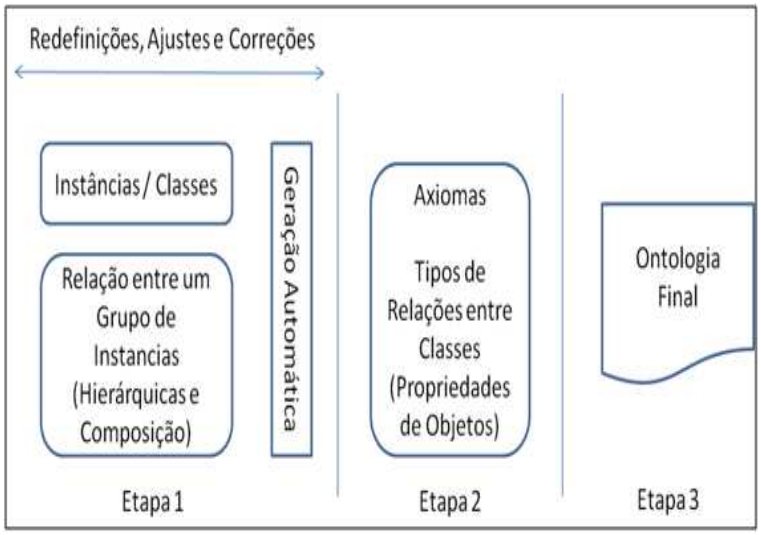

Figura 3 – Processo de Codificação de Ontologia Baseado em Instâncias. (adaptado de JORGE, 2010)

A codificação baseada no MOBI tem como base a definição de um grupo de referência, ou seja, um subconjunto de relações entre instâncias, a fim de determinar regras de

relacionamentos que irão funcionar para a composição de dos relacionamentos entre todas as instâncias, ao invés de classes. Essa mudança objetiva suavizar a rigidez dos métodos convencionais, afinal nos métodos convencionais para a mudança de uma regra existe a necessidade de mudança na estrutura da propriedade levando a modificações de uma restrição de cardinalidade.

Outro ponto é a necessidade de vincular uma instância com varias classes, no momento em que uma instância pode exercer mais de um papel. No MOBI cada instância é única, o que favorece ao paradigma de que uma instância pode ser vinculada a mais de uma classe, assim sendo pode-se demonstrar que suas relações e classificações podem mudar ao longo do seu ciclo de vida.

A arquitetura do MOBI é dividida em três níveis:

- Primeira camada – conceitual, é a especificação do MOBI.
- Segunda camada – camada de implementação, trata do projeto e implementação do MOBI, nessa camada estão a implementação do kernel e de um editor gráfico para modelagem.
- Terceira camada – representa as linguagens formais que poderão ser mapeadas, por exemplo: MOBI-OWL.

Jorge (2012) resume em dois os aspectos que nortearam o projeto MOBI: (i) elementos que diminuem o esforço da curva de aprendizado do método; (ii) processos que respeitem a dinâmica da evolução e de mudanças de uma Ontologia.

2 O Projeto do Editor de Ontologias MOBI

Este projeto propõe-se a realizar a especificação e construção de um protótipo de um editor gráfico de Ontologias baseado no MOBI. O protótipo tem a finalidade de trazer ao usuário uma maior facilidade de interação com o motor de inferência que decodifica a informação inserida pelo usuário na modelagem de determinado domínio para convertê-la, por exemplo, em um arquivo OWL. A maior facilidade decorre do fato de que não será mais necessário usar código Java para descrever a modelagem de um domínio, passando a ser feita a inserção a partir de componentes visuais que criam um ambiente mais amigável para o uso da ferramenta.

O desenvolvimento desse projeto tem como premissa conseguir um fraco acoplamento entre a interface gráfica e o kernel do MOBI, isso será conseguido através da introdução de uma camada de comunicação entre os dois módulos, ou seja, o uso de uma IDL (Linguagem de Descrição de Interface). A IDL realiza a decodificação das ações gráficas, transformando-as em métodos escritos na linguagem de programação Java que são compreendidos pelo kernel do MOBI.

2.1 Projeto arquitetural de alto nível e uma pequena descrição

O projeto de construção do protótipo do editor gráfico esta baseado na arquitetura constante na figura 4:

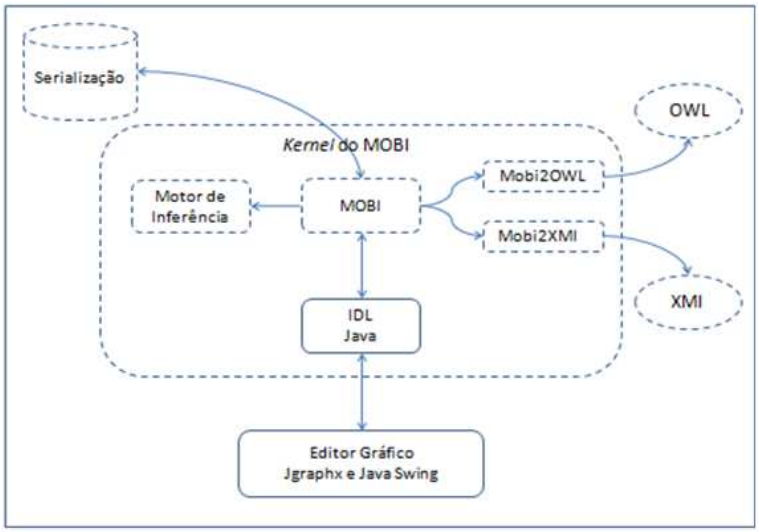

Figura 4 – Arquitetura da construção do protótipo do editor gráfico.

O diagrama arquitetural apresentado na figura 4 tem seus elementos visualmente apre-

sentados com dois traçados distintos, onde os itens em linha pontilhada fazem parte do MOBI e os itens em linha simples são as partes integrantes da especificação/implementação do Editor MOBI. Esse diagrama pode ser descrito da seguinte forma:

O bloco do kernel do MOBI representa o software de modelagem de ontologia baseado em instâncias, especificado para o método MOBI, proposto por (JORGE, 2012) e especificado e implementado na linguagem Java. Os blocos Mobi2OWL e Mobi2XML são responsáveis pela geração do produto do processamento da modelagem de um domínio a partir do MOBI, gerando respectivamente Ontologias em linguagem OWL e na notação XML.

O bloco do Motor de Inferência é o responsável pelo processamento das informações. É o motor de inferência que ira consumir as informações a cerca do domínio modelado inseridas pelo usuário.

O bloco IDL é o responsável pela interação entre a aplicação gráfica e o MOBI, essa interação é realizada através do processamento dos itens adicionados na interface gráfica obtendo como produto a inserção da informação no kernel para o processamento do motor de inferência.

O bloco Editor Gráfico é responsável por toda interação do usuário com o kernel do MOBI. Nele o usuário ira poder construir suas modelagens de dados baseadas no método MOBI apenas interagindo com objetos gráficos, não tendo a necessidade do conhecimento de nenhuma linguagem de programação.

2.2 Forma de trabalho

Como formar de trabalho o projeto foi dividido em fases. Em cada uma dessas fases foi obtido um resultado final, que foi necessário para a completude do protótipo. São elas:

Fase 1 – analise do kernel do MOBI, a fim de compreender o seu funcionamento, para daí entender o que é necessário para que a interface gráfica consiga interagir com o kernel. Neste ponto foi criado um exemplo prático de modelagem de dados sobre o método MOBI, na linguagem Java, que é a linguagem compreendida pelo kernel. Essa fase do projeto mostrou o que seria necessário para a construção da já referida IDL.

Fase 2 - consiste no desenvolvimento do módulo gráfico. Nesse módulo busca-se conseguir uma interface gráfica simples e intuitiva, com um alto grau de facilidade na interação entre o agente modelador e a ferramenta gráfica.

O desenvolvimento desse modulo irá ter por base o protótipo de tela mostrado na figura 5, este protótipo exemplifica como deve ser a tela do editor de ontologias especificado

nesse trabalho. No item 5.3 deste trabalho, será abordado o funcionamento do "Editor M-MOBI"através de uma prototipação funcional.

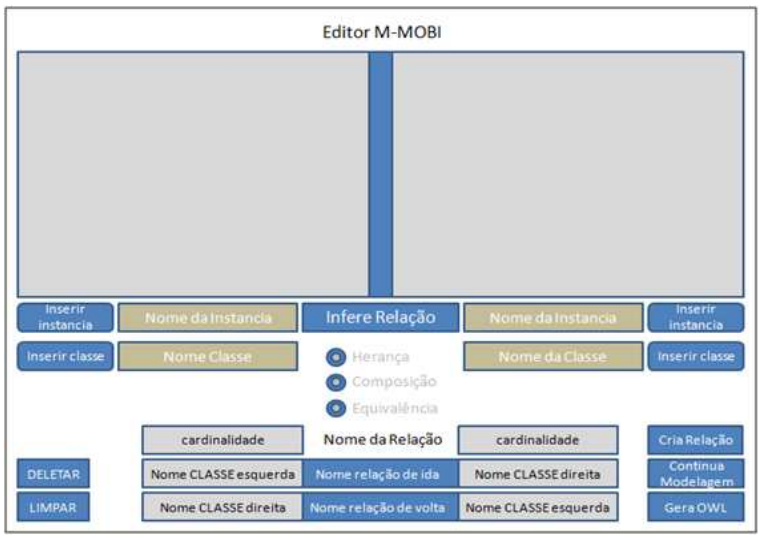

Figura 5 – Protótipo de tela.

Fase 3 – realização da junção entre a interface gráfica e o kernel, esse comunicação foi feita por meio da criação de uma IDL (Linguagem de Descrição de Interface), com a definição e criação dessa IDL tornou-se possível, que tanto a aplicação gráfica proposta e construída nesse trabalho, consiga se comunicar com o kernel do MOBI como também qualquer outra aplicação gráfica que venha a ser construída futuramente, bastando apenas existir a comunicação com a IDL de acesso ao kernel.

Fase 4 – consiste em validar o produto desenvolvido nesse trabalho. A validação será realizada a partir da modelagem de uma ontologia usando a protótipo do editor gráfico, depois essa mesma ontologia será modelada sem o uso da interface gráfica, nos dois casos será gerado um arquivo OWL como produto da modelagem realizada. Esses arquivos serão importado para o software Protégé e então seus detalhes serão comparados. Caso não exista diferença entre eles fica verificado que o protótipo do "Editor Mobi"funcionou da maneira esperada, pois conseguiu traduzir todos os componentes gráficos adicionados pelo usuario na modelagem para código java compreendido pelo kernel do MOBI.

2.3 Protótipo Funcional

Com a conclusão da especificação e implementação do editor MOBI, espera-se que a modelagem de um domínio aconteça como será mostrado em um protótipo de telas funcional apresentado nesse tópico.

Inicialmente é necessário saber o que representa cada item da interface gráfica.

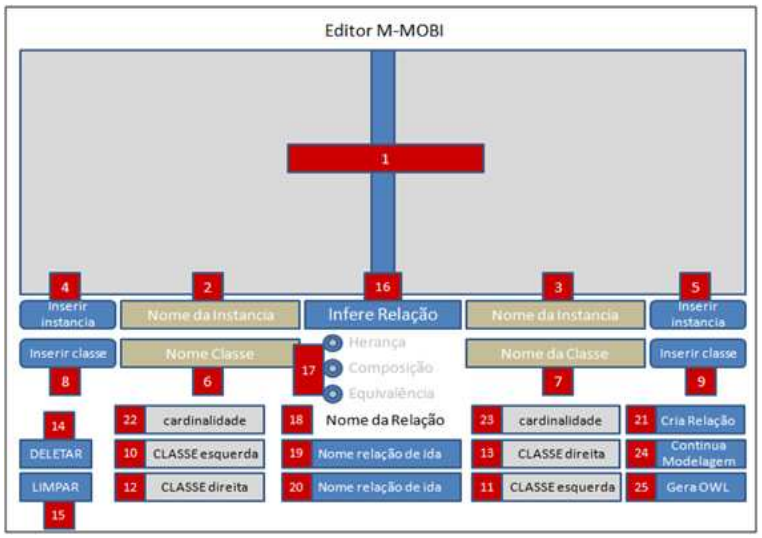

Figura 6 – Itens da interface gráfica numerados.

Na figura 6 podemos obervar todos os itens numerados. Essa numeração segue a ordem em que os elementos da interface gráfica serão usados em uma modelagem.

Item 1 – Painel onde serão exibidas as representações gráficas das instâncias e seus relacionamentos, é importante ressaltar que a linha em azul que divide o painel em dois quadrantes é meramente ilustrativa e não existira no protótipo desenvolvido.

Itens 2 e 3 - Campos em branco disponíveis para o agente modelador identificar instância que será inserida.

Itens 4 e 5 – Botões que adicionam instâncias na modelagem, essas serão exibidas no painel (área 1).

Itens 6 e 7 – Campos em branco disponíveis para o agente modelador colocar o nome da classe que será inserida.

Itens 8 e 9 – Botões que adicionam as classes.

Itens 10 e 11 – Áreas em branco onde será exibido o nome da classe inserida no lado esquerdo do editor, após o clique no botão adicionar classe do lado esquerdo.

Itens 12 e 13 – Áreas em branco onde será exibido o nome da classe inserida no lado direito do editor, após o clique no botão adicionar do lado direito.

Item 14 – Para deletar uma instância basta clicar sobre o botão delete e depois sobre a instância que deseja deletar.

Item 15 – Limpa todos os itens da tela e volta o editor ao estado inicial.

Item 16 – Botão que faz a inferência da relação. Esse botão só deve ser acionado após a adição das instâncias e das classes.

Item 17 – Um checkbox que mostrará o tipo de relação retornado pelo infere relação.

Item 18 – Apenas um label que indica que abaixo estão os campos para o agente modelador adicionar os nomes das relações no sentido de ida e volta.

Itens 19 e 20 - Campos em branco disponíveis para o agente modelador identificar a rela-

ção de ida e volta;
Item 21 – Botão que cria a relação que foi inferida.
Itens 22 e 23 – Áreas em branco onde será exibida a cardinalidade da relação nos dois sentidos.
Iten 24 – Botão que possibilita que o agente modelador limpe a tela e continue modelando o domínio em questão.
Item 25 – gera o arquivo OWL referente ao domínio modelado.

A partir da noção inicial dada sobre o funcionamento podemos adentrar na prototipação funcional. Será mostrada a modelagem da relação entre as classes "Partido" e "Candidato" do domínio eleição seguindo o paradigma do MOBI. Essa demonstração tem a finalidade de mostrar como dar-se-á o funcionamento do editor.

Para modelar através do método MOBI é necessário começar pensando em quais seriam as instâncias possíveis dentro domínio, na figura 7 esta é possível visualizar as instâncias adicionadas no quadrante esquerdo.

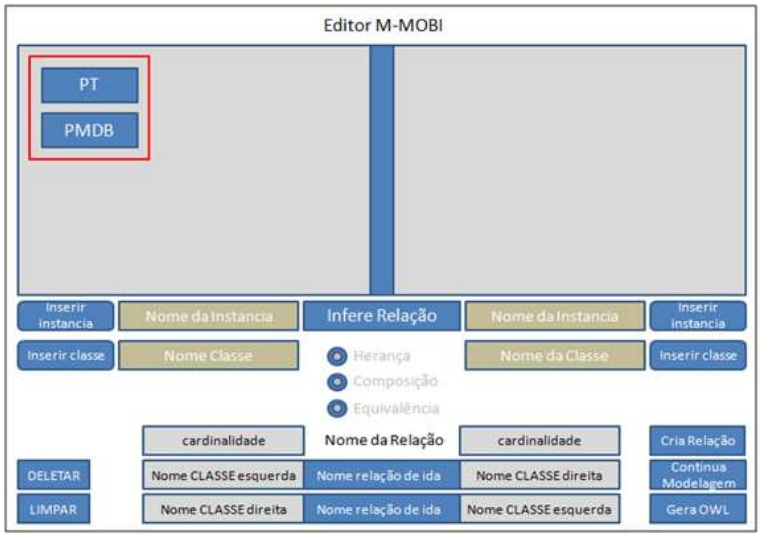

Figura 7 – Instâncias adicionadas no quadrante esquerdo.

Agora são adicionadas as instancias no quadrante direito, representado na figura 8.

Tanto para a adição de instâncias no quadrante esquerdo como no quadrante direito o agente modelador terá que nomea-las no campo "Nome Instância" e depois clicar sobre o botão "Adicionar Instância" no respectivo lado em que deseja adicionar a instância.

Na figura 9 são apresentados os relacionamentos entre as instâncias. Para criar um relacionamento basta selecionar a instância de origem e arrastar o mouse até a instância de destino.

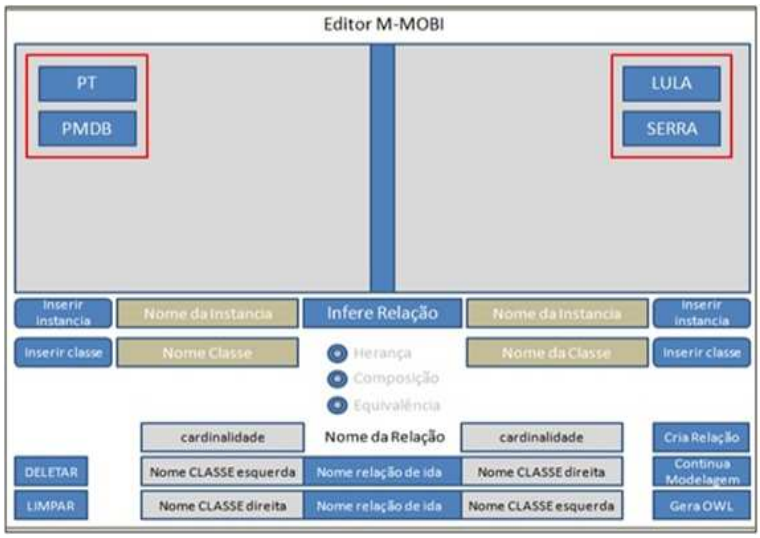

Figura 8 – Adicionadas as instâncias nos quadrantes da direita e da esquerda.

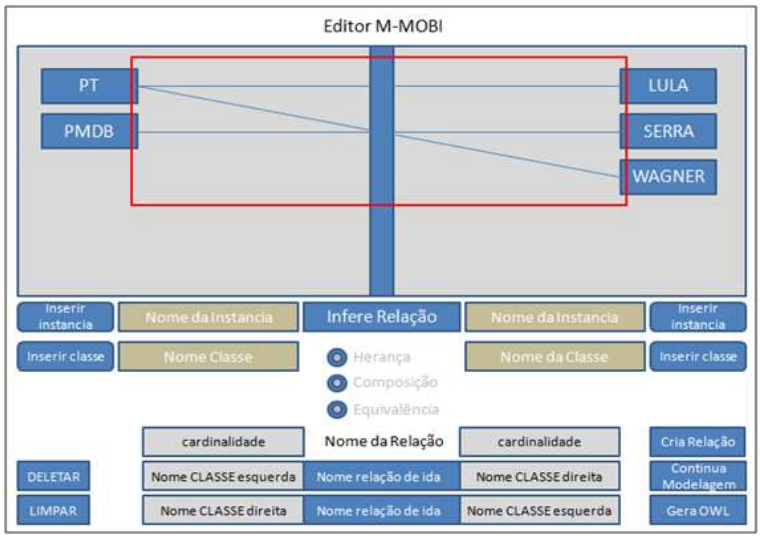

Figura 9 – Relacionamentos criados.

No próximo passo, evidenciado na figura 10 estão sendo adicionados os nomes das classes.

O agente modelador adiciona as classes, identificando-as nos campos “Nome Classe” e adicionando-as no botão “Inserir Classe”. Os nomes das classes serão exibidos também na indicação da relação. O campo “Nome da Relação” deve ser preenchido pelo agente modelador. Ver na figura 11.

Na figura 12 é evidenciado o tipo da relação que é obtido a partir do acionamento do botão infere relação. Ao acionar o botão infere relação os itens que foram adicionados na modelagem são enviados ao MOBI que, de forma quase que automática, faz a inferência do tipo da relação.

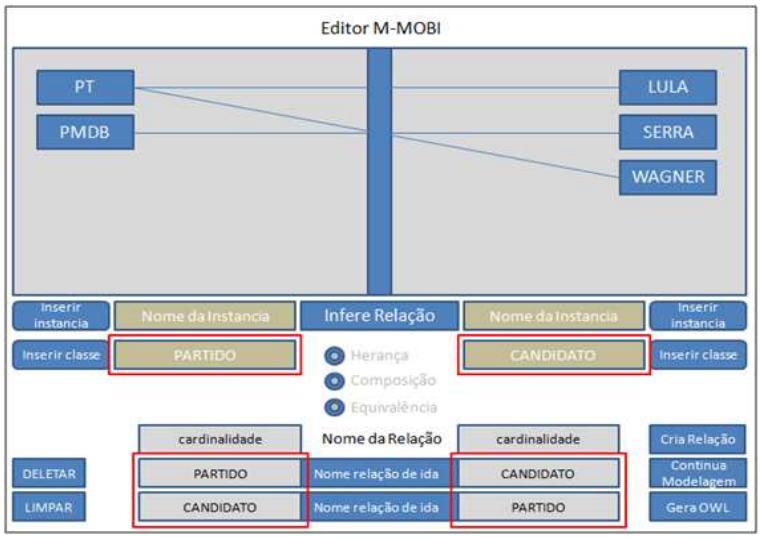

Figura 10 – Adicionando os nomes das classes.

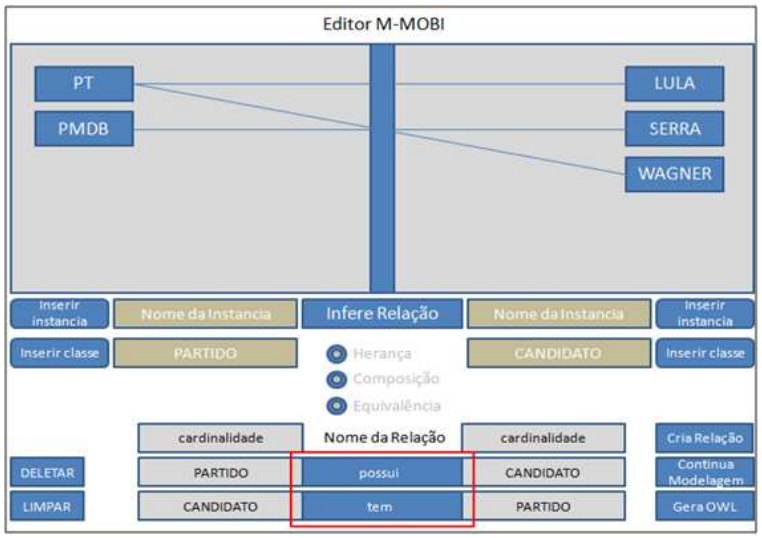

Figura 11 – Adicionando os nomes da relação.

A partir do momento que a relação foi inferida o agente modelador deverá criar a relação no MOBI, esse procedimento acontece ao acionar o botão “Cria Relação”. Ao criar a relação o MOBI irá retornar a cardinalidade da relação que será exibida nos campos cardinalidade. Como pode ser visto na figura 13.

Após criar a relação o agente modelador terá duas opções: continuar modelando ou gerar OWL. Ambas as opções podem ser iniciadas com um clique sobre o respectivo botão. Caso seja acionado o botão “Continua Modelagem” a tela será completamente limpa, para que o agente modelador possa criar uma nova relação. Caso seja acionado o botão “Gera OWL” então o MOBI irá gerar um arquivo no formato OWL produto do processamento da modelagem através do método MOBI.

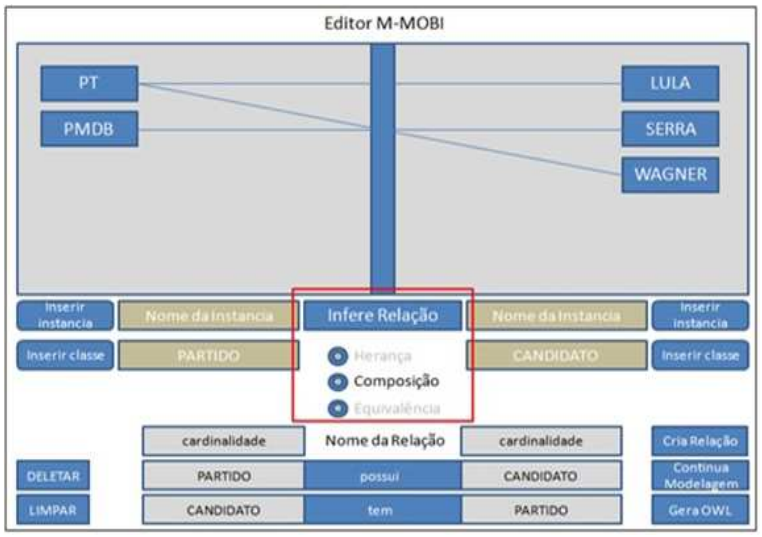

Figura 12 – Devolução do tipo de relacionamento.

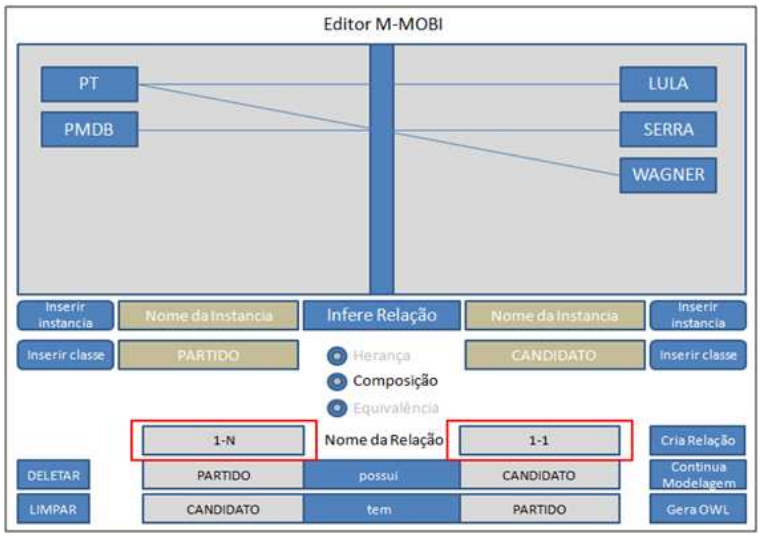

Figura 13 – Exibição da cardinalidade da relação.

2.4 Ferramentas de Desenvolvimento

No desenvolvimento do modulo gráfico estão sendo usadas as bibliotecas JGraphx e Swing, ambas são ferramentas da linguagem Java que será usada em todo o projeto. Jgraphx é uma biblioteca gráfica Java, essa biblioteca prove recursos destinados a aplicações que exibem diagramas e gráficos interativos, como por exemplo, editores gráficos. Java Swing é uma API que fornece um conjunto de elementos básicos para a construção de GUIs como, por exemplo: janelas, botões, menus, ícones, barras de rolagem, etc.

A biblioteca JGraphx esta sendo usada para a apresentação das instâncias e das relações. Graficamente as instâncias são representadas por retângulos e as relações por linhas simples. Já a API Java Swing esta sendo usada na parte estática da interface gráfica, ela

é a responsável pela exibição de campos texto, botões e demais componentes que não são criados dinamicamente.

A IDL (Linguagem de Descrição de Interface) foi construída em linguagem java e funciona como o elo entre a interface gráfica e o kernel, fazendo a conversão dos objetos gráficos para objetos entendidos pelo kernel.

Abaixo, a figura 14 que exibe a tela do protótipo do Editor Gráfico MOBI, onde podem ser observadas a parte estática composta por botões, combobox e campos texto, que foi implementada usando Java Swing e a parte dinâmica, ou seja o painel onde as instâncias são adicionadas em um processo dinamico de interação com o agente modelador e onde serão criados os relacionamentos, para isso foi usada a biblioteca Jgraph.

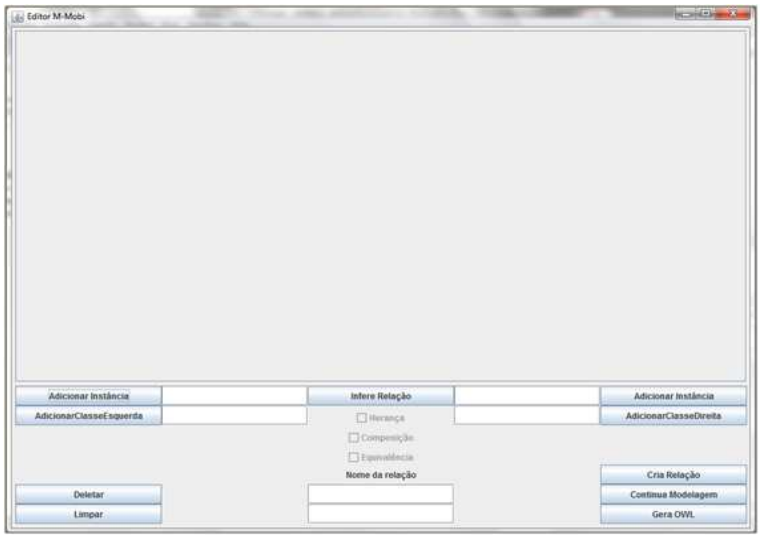

Figura 14 – Tela do editor Gráfico.

2.5 Editor MOBI na Prática e Detalhes do Desenvolvimento

Para facilitar ao agente modelador o uso correto do Editor MOBI, ele deve seguir as orientações que foram passadas acerca do funcionamento do Editor no item 2.3 Protótipo Funcional deste capítulo. Neste item será mostrado o desenvolvimento de um exemplo prático de modelagem. A modelagem acontecerá sobre o domínio Eleição. Como esse exemplo tem o objetivo de servir apenas de demonstração do correto funcionamento da aplicação, será apenas modelada a relação entre as classes “Canditato” e “Partido”.

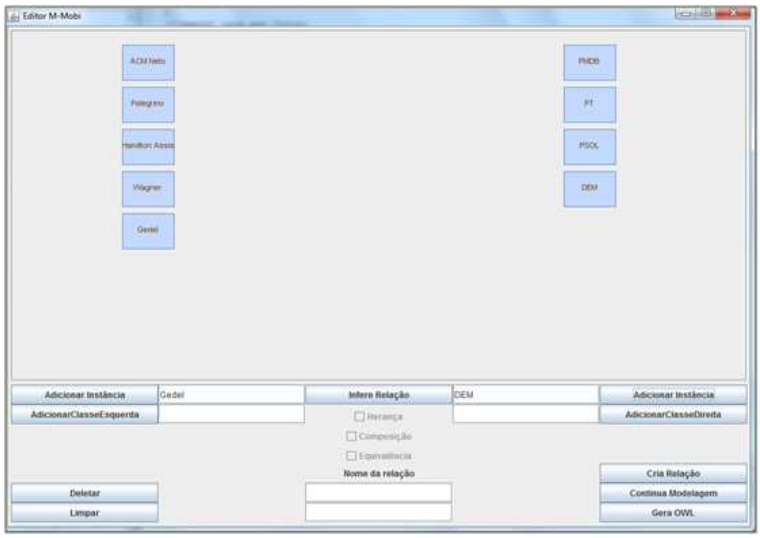

Figura 15 – Instâncias Adicionadas

Na figura 15 podem ser visualizadas as instância já adicionadas.

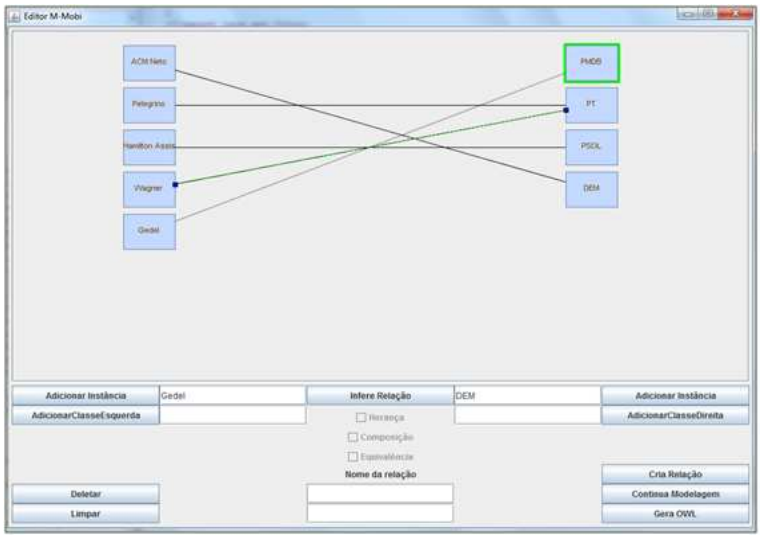

Figura 16 – Relacionamentos adicionados a modelagem

A figura 16 evidencia a criação dos relacionamentos. Existe na imagem uma linha pontilhada com detalhe em verde, essa linha é a linha do penultimo relacionamento criado que ainda estava em destaque, outro fato a ser observado é que a instância “PMDB” esta destacada em verde, isso acontece pois neste momento estava sendo finalizada a ligação ou relacionamento entre as instancias “Gedel” e “PMDB”. Após ser finalizado a ligação entre as instancias o Editor exibirá o destaque pontilhado com detalhe em verde sobre o relacionamento recém criado como pode ser visualizado na figura 17 abaixo.

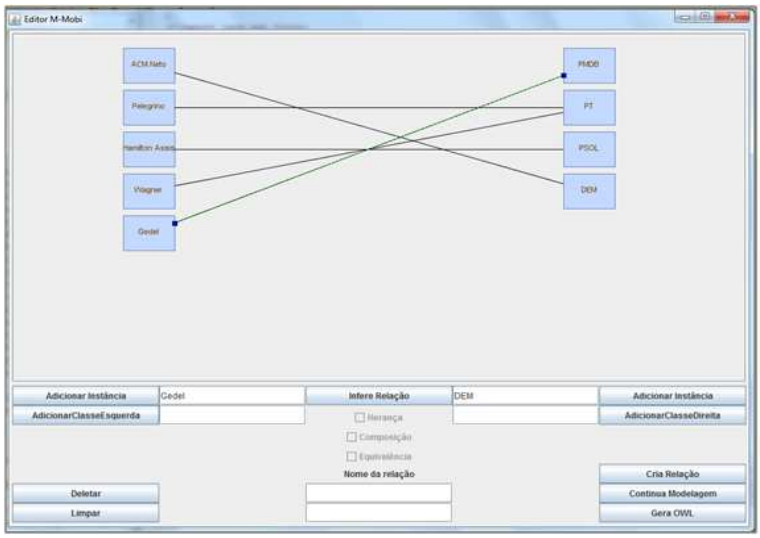

Figura 17 – Relacionamento recém definido na interface gráfica

Após definir todas as relações entre as instâncias o agente modelador precisa identificar a que classes pertencem às instâncias que foram adicionadas a modelagem. Bastando para isso adicionar os nomes das classes e clicar sobre o respectivo botão de adicionar classe. Na figura 18 as classes já foram definidas.

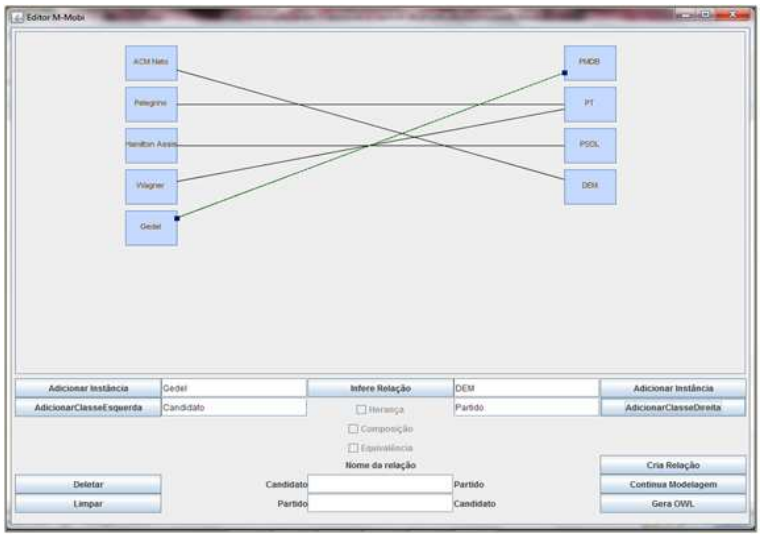

Figura 18 – Classes definidas

Após definir instâncias, relacionamentos e classes o agente modelador deve iniciar o processo criação da relação, que começa com a inferência da relação, usando o botão "Infere Relação". Esse é um ponto muito importante, pois o retorno correto do tipo de relacionamento é vital para o perfeito funcionamento da ferramenta. Uma observação importante a ser feita é dizer que o editor ainda não dá suporte para o tipo de relação

equivalência, porém já foi criado pensando em abarcar esse tipo de relação em melhorias futuras, por isso já existe na tela o checkbox Equivalência.

No momento de infererir o tipo da relação serão executados os princípais métodos do da aplicação, esses métodos fazem parte da IDL. Neste momento é iniciada a interação do Editor MOBI com o kernel. Abaixo são apresentados os principais métodos na sequência que são utilizados pelo "Editor MOBI".

O método setInstancia cria as instâncias no kernel;

```
1    public void setInstanciaMobi(List<String> nomeInstEsq, List<
        String> nomeInstDir){
2         for (int i = 0; i < nomeInstEsq.size(); i++) {
3             Instance iInstanciaGenrica = new Instance(nomeInstEsq
                 .get(i));
4             iList.add(iInstanciaGenrica);
5         }
6         for (int i = 0; i < nomeInstDir.size(); i++) {
7             Instance iInstanciaGenrica = new Instance(nomeInstDir
                 .get(i));
8             iList.add(iInstanciaGenrica);
9         }
10   }
```

Depois das instâncias serem setadas, acontece a criação do conceito das instâncias, que é feita pelo método adicionaConceitoInstancia.

```
1    public void adicionaConceitoInstancia(List<String> instEsq,
        List<String> instDir)
2         throws Exception{
3         Instance iTemp;
4         for (int j = 0; j < instEsq.size(); j++) {
5             for (int i = 0; i < iList.size(); i++) {
6                 if(instEsq.get(j).equals(iList.get(i).getUri())){
7                     iTemp = iList.get(i);
8                     mobi.addConcept(iTemp);
9                 }
10            }
11        }
12        for (int j = 0; j < instDir.size(); j++) {
13            for (int i = 0; i < iList.size(); i++) {
14                if(instDir.get(j).equals(iList.get(i).getUri())){
15                    iTemp = iList.get(i);
```

```
16                      mobi.addConcept(iTemp);
17                  }
18              }
19          }
20      }
```

Ao final do adicionaConceitoInstancia as instâncias estão criadas no MOBI. Então começa o processo de criação das classes no MOBI com a execução do método setClasseMobi, abaixo:

```
1       public void setClasseMobi(String nomeClasse){
2           Class cGenerica = new Class(nomeClasse);
3           cList.add(cGenerica);
4           nomeClasses.add(nomeClasse);
5       }
```

Após a execução do setClasseMobi é necessário adicionar conceito as classes

```
1       public void adicionaConceitoClasse(String nomeClasseEsq,
           String nomeClasseDir) throws Exception{
2           Class cTemp;
3           for (int j = 0; j < cList.size(); j++) {
4               if(nomeClasseEsq.equals(cList.get(j).getUri())){
5                   cTemp = cList.get(j);
6                   mobi.addConcept(cTemp);
7               }
8           }
9           for (int j = 0; j < cList.size(); j++) {
10              if(nomeClasseDir.equals(cList.get(j).getUri())){
11                  cTemp = cList.get(j);
12                  mobi.addConcept(cTemp);
13              }
14          }
15      }
```

Agora com classes e instâncias devidamente criadas é necessário informar ao MOBI a qual classe cada instancia pertence, isso é feito no método instanciaDaClasse.

```
1       public void instanciaDaClasse(
2           String nomeClasseEsq, String nomeClasseDir, List<String>
              instEsq, List<String> instDir)
3           throws ExceptionURI{
4           Class cTempEsq = mobi.getClass(nomeClasseEsq);
5           Class cTempDir = mobi.getClass(nomeClasseDir);
```

```
6           Instance iTempEsq = null;
7           Instance iTempDir = null;
8           if(instEsq != null){
9               for (int i = 0; i < instEsq.size(); i++) {
10                  iTempEsq = mobi.getInstance(instEsq.get(i));
11                  mobi.isOneOf(iTempEsq, cTempEsq);
12              }
13          }
14          if(instDir != null){
15              for (int i = 0; i < instDir.size(); i++) {
16                  iTempDir = mobi.getInstance(instDir.get(i));
17                  mobi.isOneOf(iTempDir, cTempDir);
18              }
19          }
20      }
```

Após o MOBI conhecer a qual classe cada instância pertence é a hora de inferir o tipo de relação, para isso é executado o método relacaoGenerica.

```
1       public Collection<Integer> relacaoGenerica(
2               String nomeClasseEsq, String nomeClasseDir, List<
                   Relacionamento> relacionamentos_,
3               List<String> instEsquerdaString_, List<String>
                   instDireitaString_) throws Exception{
4           String instEsq, instDir;
5           List<String> relacionamentoInstancias = new ArrayList<
               String>();
6
7           Class cTempEsq = mobi.getClass(nomeClasseEsq);
8           genericRelation.setClassA(cTempEsq);
9           Class cTempDir = mobi.getClass(nomeClasseDir);
10          genericRelation.setClassB(cTempDir);
11
12          for (int l = 0; l < relacionamentos_.size(); l++) {
13
14              instEsq = relacionamentos_.get(l).
                   getInstanciaEsquqerda();
15              instDir = relacionamentos_.get(l).getInstanciaDireita
                   ();
16              genericRelation.addInstanceRelation(mobi.getInstance(
                   instEsq), mobi.getInstance(instDir));
17          }
```

```
18
19          List<String> instaciaNaoTemRelacionamentoEsq = new
               ArrayList<String>();
20          List<String> instaciaNaoTemRelacionamentoDir = new
               ArrayList<String>();
21
22          for (int i = 0; i < relacionamentos_.size(); i++) {
23              String temp;
24              temp = relacionamentos_.get(i).getInstanciaEsquqerda
                   ();
25              relacionamentoInstancias.add(temp);
26              temp = relacionamentos_.get(i).getInstanciaDireita();
27              relacionamentoInstancias.add(temp);
28          }
29          for (int m = 0; m < instEsquerdaString_.size(); m++) {
30              if( ! relacionamentoInstancias.contains(
                   instEsquerdaString_.get(m)) ){
31                  instaciaNaoTemRelacionamentoEsq.add(
                       instEsquerdaString_.get(m));
32              }
33          }
34          for (int m = 0; m < instDireitaString_.size(); m++) {
35              if( ! relacionamentoInstancias.contains(
                   instDireitaString_.get(m)) ){
36                  instaciaNaoTemRelacionamentoDir.add(
                       instDireitaString_.get(m));
37              }
38          }
39          if (instaciaNaoTemRelacionamentoEsq.size() > 0) {
40              for (int n = 0; n < instaciaNaoTemRelacionamentoEsq.
                   size(); n++) {
41                  instEsq = instaciaNaoTemRelacionamentoEsq.get(n);
42                  genericRelation.addInstanceRelation(mobi.
                       getInstance(instEsq), null);
43              }
44          }
45          if (instaciaNaoTemRelacionamentoDir.size() > 0) {
46              for (int n = 0; n < instaciaNaoTemRelacionamentoDir.
                   size(); n++) {
47                  instDir = instaciaNaoTemRelacionamentoDir.get(n);
48                  genericRelation.addInstanceRelation(mobi.
```

```
                    getInstance(instDir), null);
49              }
50          }
51          genericRelation.processCardinality();
52          mobi.addConcept(genericRelation);
53          Collection<Integer> possibilities = mobi.infereRelation(
                genericRelation);
54          return possibilities;
55      }
```

O método relacaoGenerica retorna os possíveis tipos, esses tipos são aplicados a uma função que define o tipo exato da relação e então o checkbox será marcado conforme pode ser visto na figura 19.

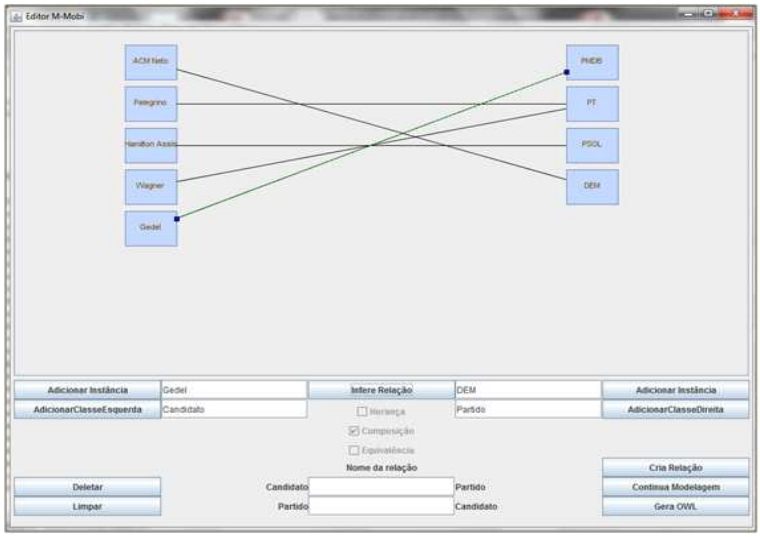

Figura 19 – Retorno do tipo de relação

Com o retorno do tipo de relacionamento inferido pelo MOBI, o editor só deixa a opção de “Criar Relação” ativa. Então o agente modelado deve criar a relação, neste momento o motor de inferência irá processar a modelagem e retornará a cardinalidade da relação. A cardinalidade da relação será exibida abaixo dos nomes das classes. Conforme a figura 20.

Figura 20 – Cardinalidade e nomes das relações definidos.

Neste ponto o agente modelador terá duas opções: a primeira é “Gerar OWL” que irá invocar a criação do arquivo OWL produto da modelagem construida e a segunda “Continuar Modelagem”. Caso a opção seja continuar a modelagem a tela será limpa para a continuação da modelagem de novas relações. Neste exemplo a opção escolhida foi a geração do arquivo OWL. Abaixo na Figura 21 a imagem do arquivo OWL gerado.

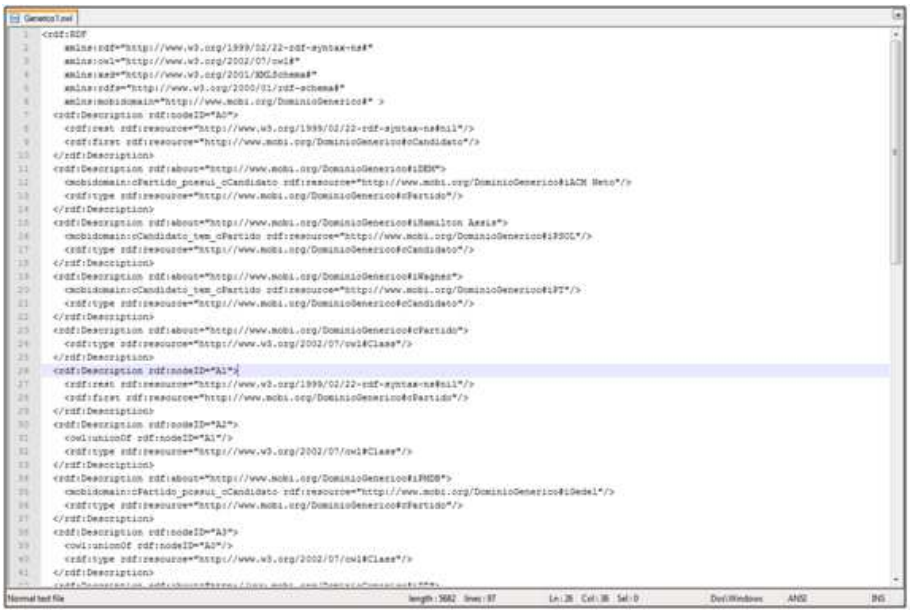

Figura 21 – Arquivo OWL produto da modelagem

2.6 Limitação do Projeto

Como já foi mensionado na seção 2.5 o “Editor MOBI” especificado/implementado neste trabalho tem a limitação do tratamento da relação do tipo Equivalência. Apesar de já existir suporte gráfico para esse tipo de relação, ela ainda não teve sua lógica de construção do relacionamento implementada por uma questão de organização das metas e prazos do projeto, por isso, foi subtraída do produto desse projeto e adicionada como uma perspectiva de futura.

3 Validação do Projeto

Para validar o sucesso da especificação e construção do “Editor MOBI” foi modelado um relacionamento do domínio eleição. Esta modelagem foi feita de duas maneira, a primeira utilizando a modelagem direta da ontologia no kernel do MOBI, criando sua descrição através da linguagem java, na forma compreendida pelo kernel e a segunda utilizando o "Editor MOBI"construido e especificado neste trabalho.

A primeira forma de modelagem resultou no código de descrição da ontologia escrito na linguagem java que esta evidenciado logo abaixo. Esse código especifica a construção da relação entre as intâncias da classes Candidato e Partido. Após o processamento deste código o kernel do MOBI respondeu com a geração do arquivo OWL, que será usado após a realização da modelagem no Editor MOBI sendo comparado com o arquivo gerado a partir da modelagem no Editor.

```
1     public class Validacao {
2         Mobi mobi = new Mobi("DominioGenerico");
3
4         public static void main(String[] args) {
5             Validacao v = new Validacao();
6             v.carregaDominioGenerico();
7         }
8
9         public void carregaDominioGenerico() {
10            try
11            {
12                Instance iACMNeto = new Instance("iACMNeto");
13                Instance iPelegrino = new Instance("iPelegrino");
14                Instance iHamilton = new Instance("iHamilton");
15                Instance iWagner = new Instance("iWagner");
16                Instance iGedel = new Instance("iGedel");
17
18                mobi.addConcept(iACMNeto);
19                mobi.addConcept(iPelegrino);
20                mobi.addConcept(iHamilton);
21                mobi.addConcept(iWagner);
22                mobi.addConcept(iGedel);
23
24                Instance iPT = new Instance("iPT");
```

```
25                  Instance iDEM = new Instance("iDEM");
26                  Instance iPMDB = new Instance("iPMDB");
27                  Instance iPSOL = new Instance("iPSOL");
28
29                  mobi.addConcept(iPT);
30                  mobi.addConcept(iDEM);
31                  mobi.addConcept(iPMDB);
32                  mobi.addConcept(iPSOL);
33
34                  Class cCandidato = new Class("cCandidato");
35                  Class cPartido   = new Class("cPartido");
36
37                  mobi.addConcept(cCandidato);
38                  mobi.addConcept(cPartido);
39
40                  mobi.isOneOf(iACMNeto, cCandidato);
41                  mobi.isOneOf(iPelegrino, cCandidato);
42                  mobi.isOneOf(iHamilton, cCandidato);
43                  mobi.isOneOf(iWagner, cCandidato);
44                  mobi.isOneOf(iGedel, cCandidato);
45
46                  mobi.isOneOf(iPT, cPartido);
47                  mobi.isOneOf(iDEM, cPartido);
48                  mobi.isOneOf(iPMDB, cPartido);
49                  mobi.isOneOf(iPSOL, cPartido);
50
51                  GenericRelation genericRelation = (
                        GenericRelation)mobi.createGenericRelation("
                        generic1");
52                  genericRelation.setClassA(cCandidato);
53                  genericRelation.setClassB(cPartido);
54                  genericRelation.addInstanceRelation(iACMNeto,
                        iDEM);
55                  genericRelation.addInstanceRelation(iPelegrino,
                        iPT);
56                  genericRelation.addInstanceRelation(iWagner, iPT)
                        ;
57                  genericRelation.addInstanceRelation(iHamilton,
                        iPSOL);
58                  genericRelation.addInstanceRelation(iGedel, iPMDB
                        );
```

```
59                  genericRelation.processCardinality();
60                  mobi.addConcept(genericRelation);
61
62                  Collection<Integer> possibilities = mobi.
                       infereRelation(genericRelation);
63                  for(Integer i: possibilities)
64                      System.out.println(i.toString());
65
66                  if(possibilities.contains(Relation.
                       BIDIRECIONAL_COMPOSITION))
67                  {
68                      CompositionRelation composition =
69                      (CompositionRelation)mobi.
                           convertToBidirecionalCompositionRelationship
                           (genericRelation, "tem", "possui");
70                      mobi.addConcept(composition);
71                  }
72
73              }catch(Exception e)
74              {
75                  e.printStackTrace();
76              }
77              geraOWL();
78          }
79          public void geraOWL(){
80              Mobi2OWL mobi2OWL = new Mobi2OWL("http://www.mobi.org
                   /", mobi);
81              mobi2OWL.setExportPath("C:\\BaseOntologia");
82              mobi2OWL.exportMobiToOWL("Validacao2Generico1.owl");
83              System.out.println("MOBI");
84          }
85      }
```

Na segunda maneira a modelagem foi realizada usando o "Editor MOBI". Ao terminar de modelar a relação a tela do editor estava conforme a figura 22. Nela podem ser visualizadas as cardinalidades da relação que são retornadas pelo kernel do MOBI após o acionamento do processo "criar relação".

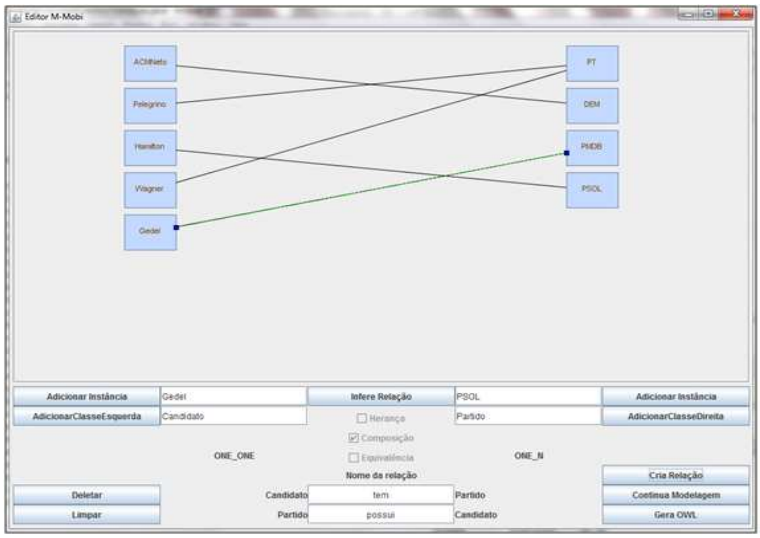

Figura 22 – Modelagem da relação entre Candidato e Partido

As cardinalidade na figura 22 indicam que um candidato tem um partido e um partido possui n candidatos. Com o retorno correto da cardinalidade da relação que foi modelada, fica evidenciado que a aplicação gráfica esta comunicando-se de maneira correta com o kernel do MOBI. Então deve ser feita a geração do arquivo OWL que descreve a relação modelada.

Uma vez que já foram gerados os dois arquivos OWL resultantes das duas formas de modelagem, os arquivos foram importados para o Protégé (plataforma que dá suporte a modelagem de Ontologias) um por vez, onde os detalhes da modelagem foram apresentados e agora serão comparados.

O primeiro arquivo OWL gerado foi o "Validacao2Generico2.owl"resultante da primeira modelagem e pode ser identificado na figura 23.

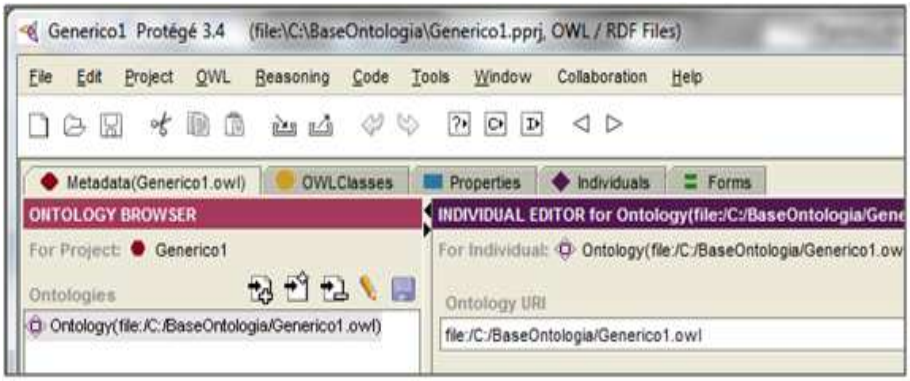

Figura 23 – Identificação do arquivo da primeira modelagem no Protége

O segundo arquivo OWL gerado foi o "Generico1.owl"resultante da segunda modelagem realizada através do "Editor MOBI"pode ser identificado na figura 24.

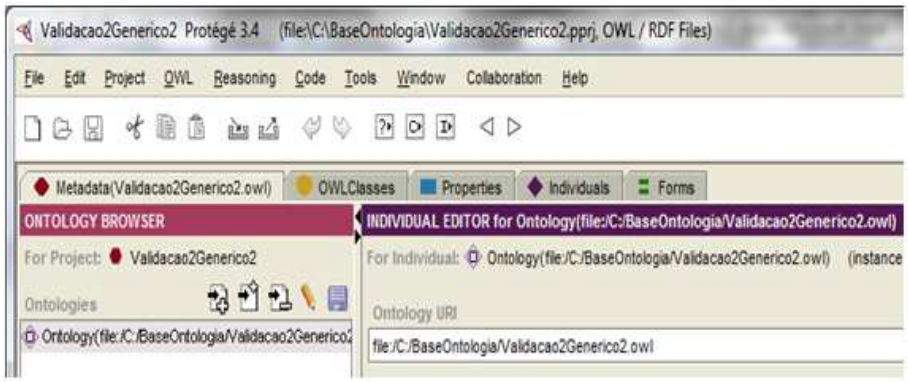

Figura 24 – Identificação do arquivo da segunda modelagem no Protége

Agora que já foram identificados os dois arquivos owl no protégé será feita a comparação em relação aos detalhes da modelagem.

Nas figura 25 e 26 podemos ver respectivamente as classes no arquivo "Validacao2Generico2.owl"e "Generico1.owl", então podemos concluir que tratam-se das mesmas classes.

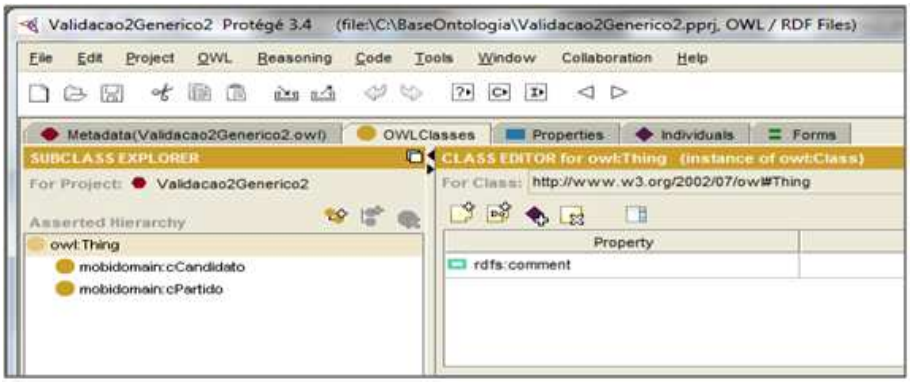

Figura 25 – Classes no arquivo Validacao2Generico2.owl

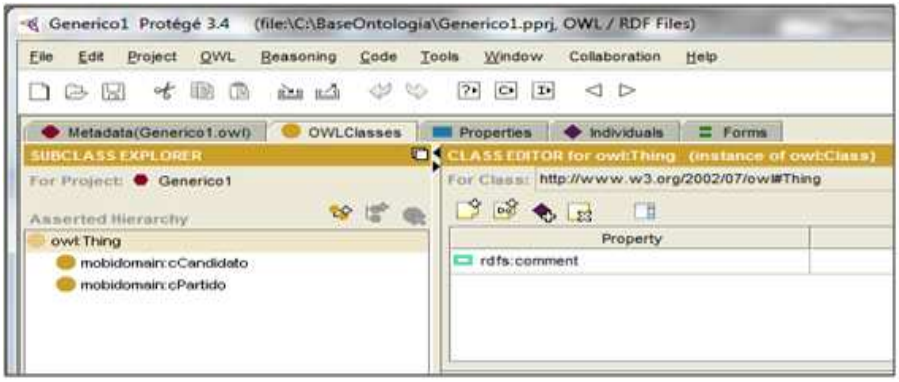

Figura 26 – Classes no arquivo Generico1.owl

Nas figuras 27 e 28 podem ser vistas as relações propriedades da relação constante no arquivo "Validacao2Generico2.owl"e nas figuras 29 e 30 constatamos que no arquivo "Generico1.owl"as propriedades são as mesmas.

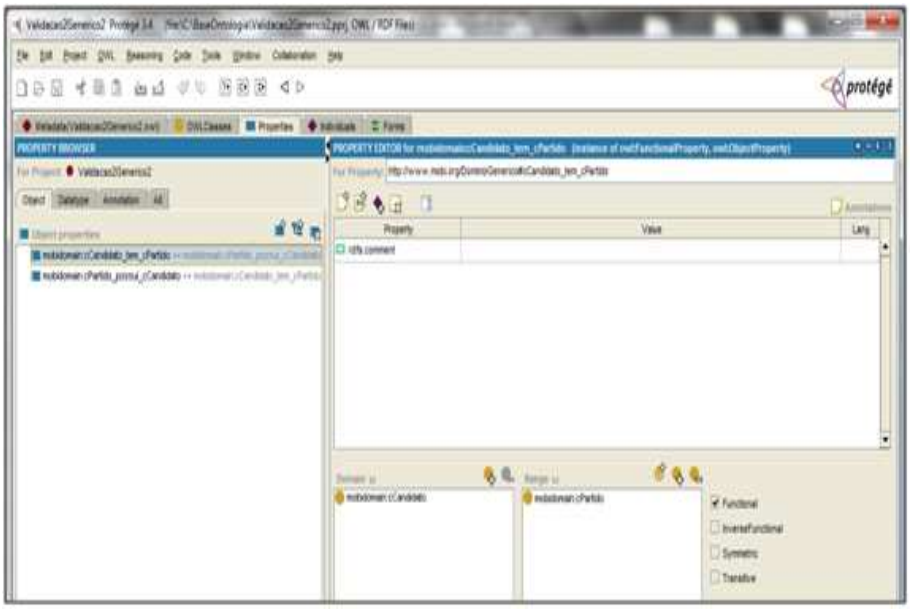

Figura 27 – Validacao2Generico2.owl propriedade funcional

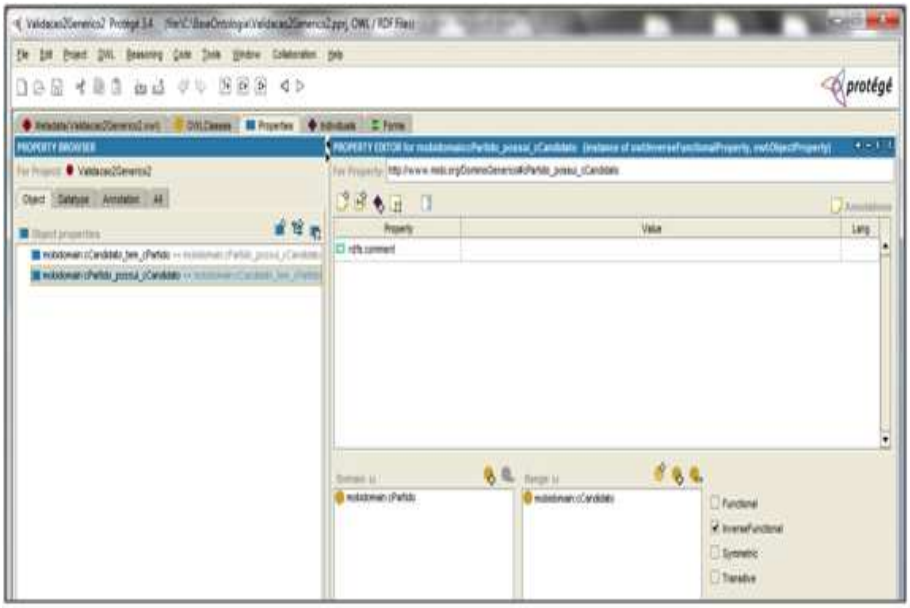

Figura 28 – Validacao2Generico2.owl propriedade inversa funcional

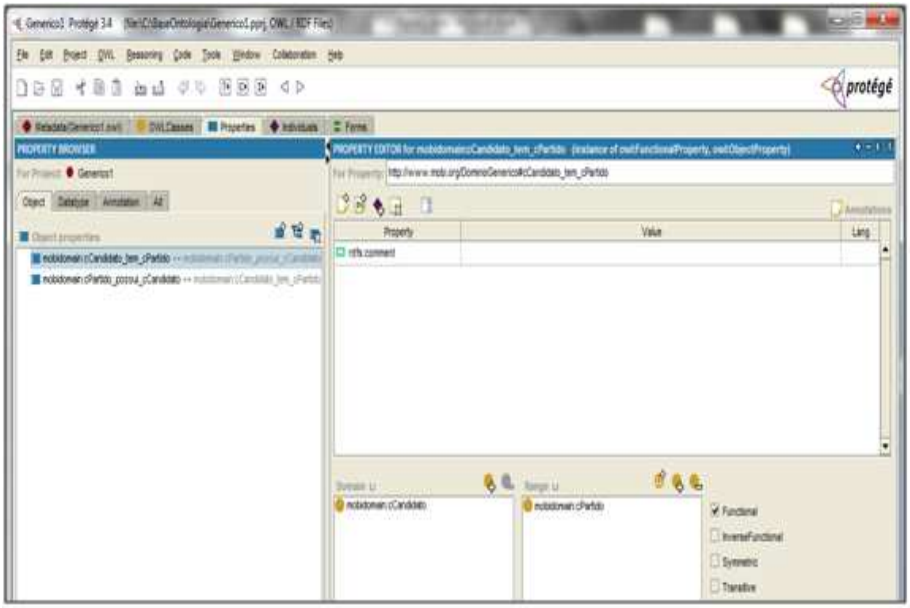

Figura 29 – Generico1.owl propriedade funcional

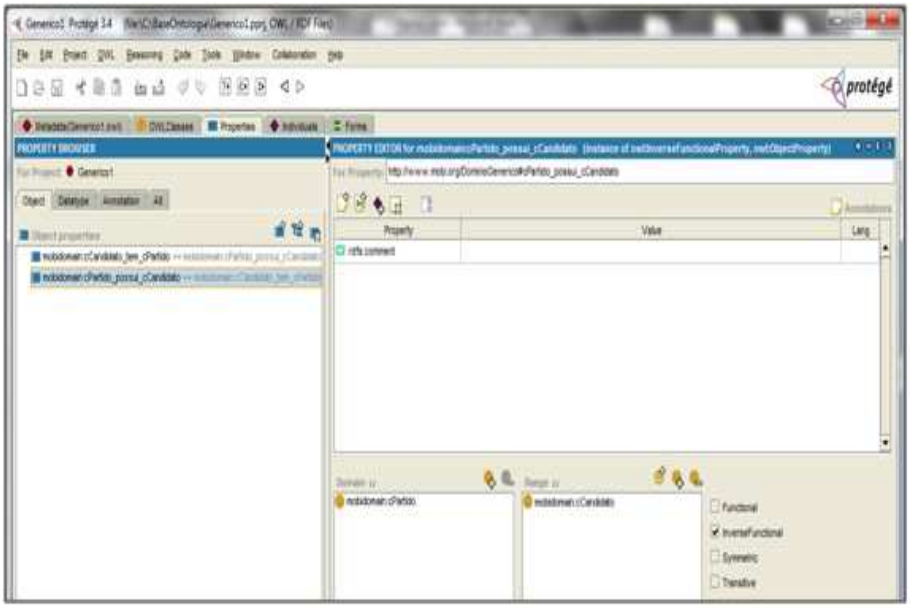

Figura 30 – Generico1.owl propriedade inversa funcional

Fica confirmado que as propriedades são as mesmas em ambas as formas de modelagem. Por fim serão mostrados as instâncias pertencentes a cada classe. As figuras figuras 31 e 32 são referentes ao arquivo "Validacao2Generico2.owl"e a figura 31 exibe as intâncias da classe "Candidato"e a figura 32 exibe as da classe "Partido". Já as figuras figuras 33 e 34 são referentes ao arquivo "Generico1.owl"e a figura 33 exibe as intâncias da classe "Candidato"e a figura 34 exibe as da classe "Partido"

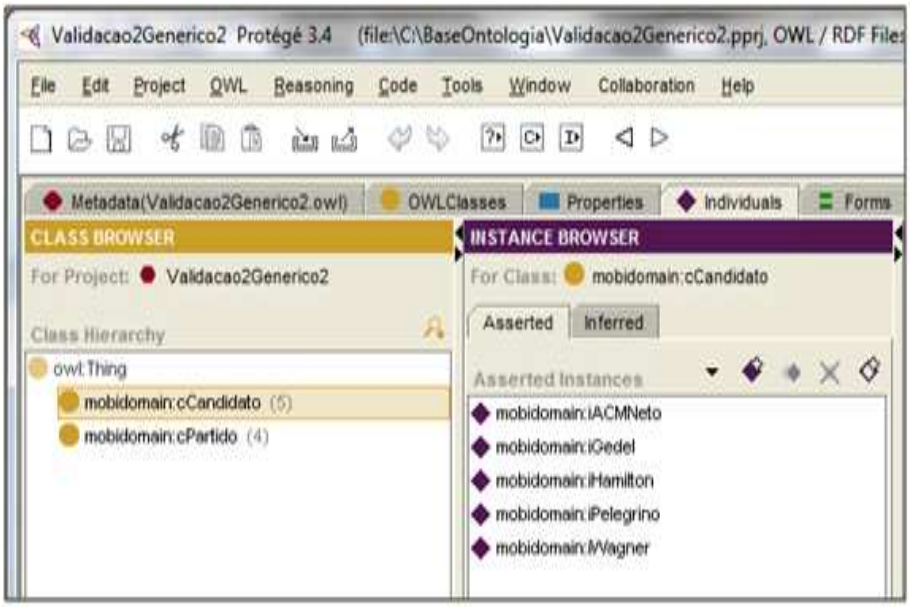

Figura 31 – Intâncias da classe "Candidato"do arquivo Validacao2Generico2.owl.

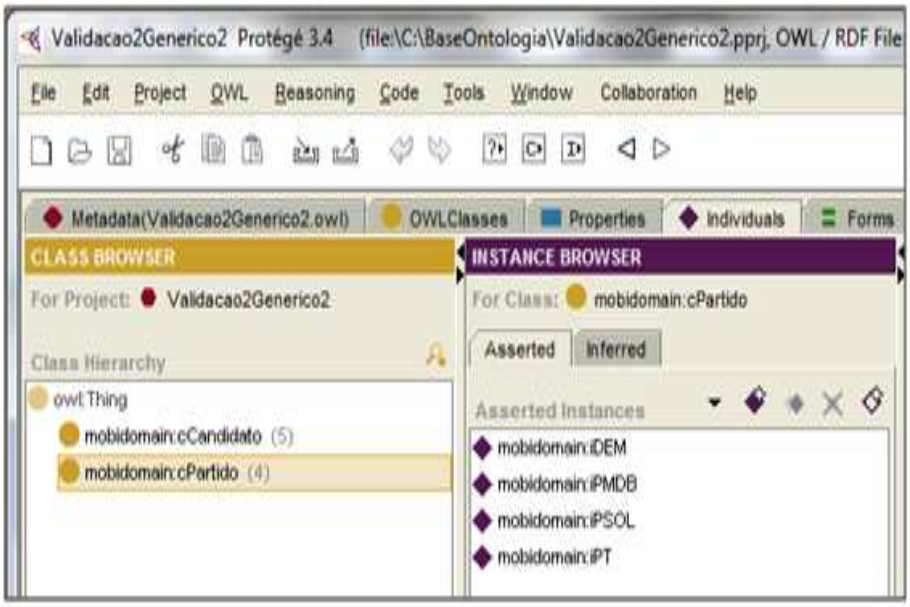

Figura 32 – Intâncias da classe "Partido"do arquivo Validacao2Generico2.owl.

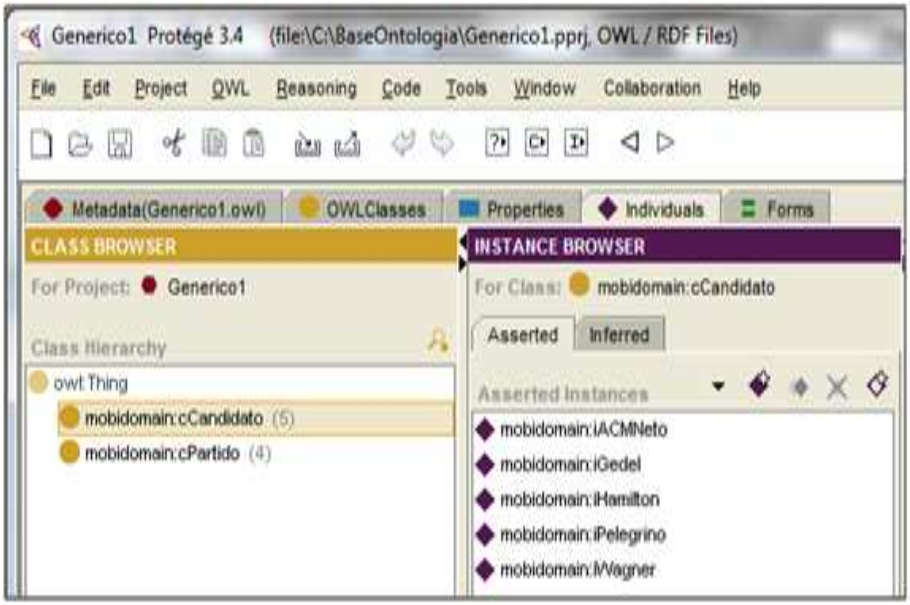

Figura 33 – Intâncias da classe "Candidato"do arquivo Generico1.owl.

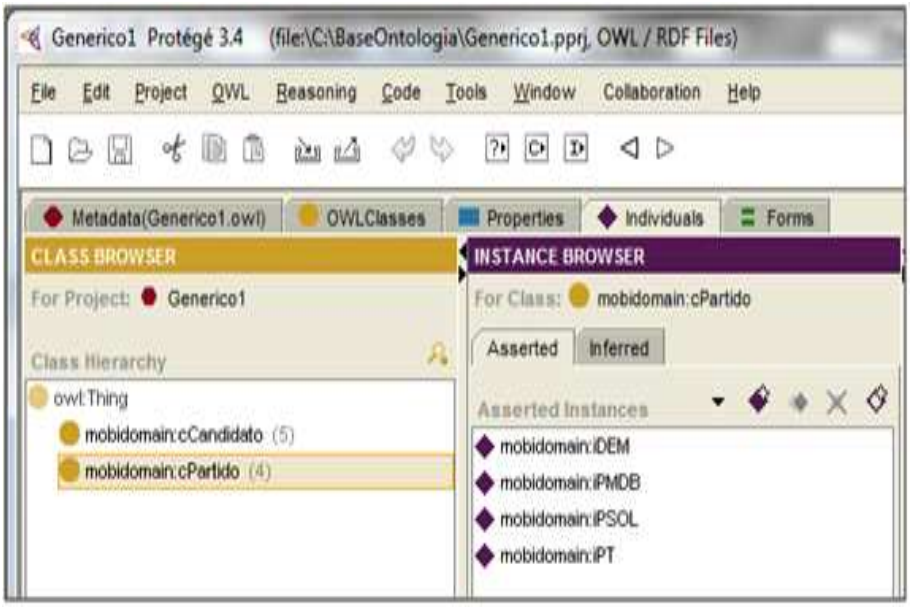

Figura 34 – Intâncias da classe "Partido"do arquivo Generico1.owl.

A partir da comparação apresentada entre os detalhes das modelagens exibidos no Protégé evidênciamos a corretude do processo de modelagem de ontologia a partir do "Editor MOBI", pois modelando a mesma relação no Editor e na forma original de modelagem do kernel, inserindo a descrição da ontologia em código java obtvemos os mesmos resultados, mostrando assim que o Editor conseguiu interagir com o kernel.

4 Considerações Finais e Recomendações

4.1 Conclusão

Devido ao vultuoso volume de informações dispersas, por exemplo, na internet e as variadas formas representação das informações, sem que a elas estejam associadas ao uso de semântica para guiar as buscas por essas informações surge a necessidade de mecanismos que possibilitem a recuperação da informação. Como ao se pensar em recuperação da informação, o tema ontologias deve ser levado em conta, em decorrência de sua capacidade de sistematizar as informações, e construir uma Ontologia não é algo trivial, necessitando de conhecimento do domínio a ser modelado e das ferramentas de modelagem de Ontologias, este trabalho situa-se na minimização da complexidade do uso e compreensão da ferramenta de modelagem de ontologia MOBI, especificando, construindo e validando o editor MOBI, que visa tornar simples o ato de modelar uma ontologia através do método MOBI.

Neste trabalho foi especificado, construído e validado o protótipo do Editor MOBI. Para alcançar o objetivo, primeiro foram pesquisados os temas relacionados, constantes na fundamentação teórica deste trabalho, são eles Ontologia, OWL e MOBI, depois foi construído um protótipo de tela que norteou o desenvolvimento da aplicação, daí aconteceram as pesquisas sobre as bibliotecas de interface gráfica que possibilitavam uma produtividade e assertividade ao que se almejava. Definidas as bibliotecas gráficas que seriam usadas, então aconteceu o desenvolvimento da interface gráfica, que uma vez concluído, gerou o iniciou do desenvolvimento da IDL que contém todo o processo de transformação dos objetos gráficos utilizados em uma modelagem em informações compreendidas pelo kernel, alcançando assim com o desenvolvimento da IDL um fraco acoplamento entre a interface gráfica e o kernel do MOBI.

Ao final desse processo foi obtida uma versão do protótipo do Editor MOBI funcional e fluida, que interage corretamente com o núcleo do kernel enviando exatamente o que foi criado na interface gráfica pelo agente modelador, atingindo uma das consequências do objetivo deste projeto, que é a não existência da necessidade do conhecimento de linguagem de programação para o uso da ferramenta MOBI. Um ponto importante para essa conclusão é o correto funcionamento do motor de inferência que está de maneira coerente processando as informações da interface gráfica, evidenciando o funcionamento correto do modulo IDL, responsável pela comunicação entre interface gráfica e o kernel.

O projeto foi concluído com êxito, pois teve como produto final uma ferramenta que

funciona dentro dos padrões esperados para a modelagem de dominios através do método MOBI.

4.1.1 Perspectivas Futuras

O Editor MOBI especificado/implementado neste trabalho pode passar por um processo de expansão, esse processo pode abarcar três itens que trariam avanços significativos para a ferramenta.

- O primeiro é o desenvolvimento do suporte a modelagem de relações do tipo "Equivalência"que como foi mensionado no item 2.6 deste projeto.
- O segundo é a especificação/implementação de painel auxiliar que exiba os relacionamentos entre classes que estão sendo criadas, afim de facilitar a interação do agente modelador com o "Editor MOBI".
- O terceiro é a expansão do editor, passando a modelar também diagramas UML e entidade relacionamento que já são produtos possíveis a partir do kernel do MOBI.

Referências

ALMEIDA, M. B. *Um modelo baseado em ontologias para representação da memória organizacional*. Tese (Doutorado) — Escola de Ciência da Informação da UFMG, 2006. 13

BREITMAN, K. *Web Semântica a Internet do Futuro*. 1. ed. Rio de Janeiro – RJ: Editora LTC, 2005. 10, 18

FOLHA de São Paulo: Site. 2013. Disponível em: <http://www1.folha.uol.com.br/mercado/2013/05/1279552-acesso-a-internet-no-brasil-cresce-mas-53-da-populacao-ainda-nao-usa-a-rede.shtml>. Acesso em: 05 nov. 2013. 9

FORTE, M.; SOUZA, W. L. de; PRADO, A. F. do. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. In: *Ciência da Informação. v. 26, n. 1. p. 39-45, 2003a*. [S.l.: s.n.], 2003. 10

FORTE, M.; SOUZA, W. L. de; PRADO, A. F. do. Utilizando ontologias e serviços web na computação ubíqua. In: *XX Simpósio Brasileiro de Engenharia de Software*. [S.l.: s.n.], 2006. 16

FORTE, M.; SOUZA, W. L. de; PRADO, A. F. do. Utilizando ontologias e serviços web na computação ubíqua. In: *XX Simpósio Brasileiro de Engenharia de Software*. [S.l.: s.n.], 2006.

GUARINO, N. Formal ontology and information systems. In: *National Research Council, LADSEB-CNR, Corso Stati Uniti 4, I-35127 Padova, Italy*. [S.l.: s.n.], 1998. 13

GUIMARãES, F. J. Z. *Utilização de ontologias no domínio B2C*. Dissertação (Mestrado) — Pontificia Universidade Católica do Rio de Janeiro, 2002. 12

HAAV, H.-M.; LUBI, T.-L. A survey of concept-based information retrieval tools on the web. In: *5th East-European Conference, ADBIS 2001*. Vilnius, Lithuania: [s.n.], 2001. 14

HEIJST, G. van; SCHREIBER, A. T.; WIELINGA, B. J. Using explicit ontologies in kbs development. *Int. J. Hum.-Comput. Stud.*, Academic Press, Inc., Duluth, MN, USA, v. 46, n. 2-3, p. 183–292, mar. 1997. ISSN 1071-5819. Disponível em: <http://dx.doi.org/10.1006/ijhc.1996.0090>. 14

JORGE, E. M. de F. *MOBI – MODELO DE ONTOLOGIA BASEADO EM INSTÂNCIA*. Tese (Doutorado) — Doutorado Multiinstitucional e Multidisciplinar em Difusão do Conhecimento da UFBA / LNCC / UNEB / UEFS / UFABC / IFBA / SENAI-CIMATEC, 2012. 10, 17, 18, 20, 22

LIMA, G. A. B. Interfaces entre a ciência da informação e a ciencia cognitiva. In: *Ci. Inf., Brasília, v. 32, n. 1, p. 77-87, jan-abr. 2003*. [S.l.: s.n.], 2003.

LIMA, J. C. de; CARVALHO, C. L. de. Relatório Técnico, *Ontologias - OWL (Web Ontology Language)*. 2005. 15, 16

MAEDCHE, A. *Ontology Learning for the Semantic Web*. [S.l.]: Kluwer Academic Publishers, 2002. 12

OYOLA, A. V.; ALVARENGA, L. Principios metodologicos interdisciplinares no processo de construção de ontologias. In: *X Encontro Nacional de Pesquisa em Ciência da Informação (ENANCIB), 25 a 28 de outubro de 2009, realizado em João Pessoa, Paraíba*. [S.l.: s.n.], 2009. 12

SILVA, D. L. D. *Um Proposta Metodológica para a Construção de Ontologias: Uma Perspectiva Interdisciplinar entre as Ciências da Informação e da Computação*. Dissertação (Mestrado) — Escola de Ciência da Informação da UFMG, 2008. 14

SILVA, D. L. da; SOUZA, R. R.; ALMEIDA, M. B. Principios metodologicos interdisciplinares no processo de construção de ontologias. In: *Seminário de Pesquisa em Ontologia no Brasil, 3º OntoBras, 30 e 31 de agosto de 2010 - Florianópolis/SC*. [S.l.: s.n.], 2010. 12

USCHOLD, M.; GRUNINGER, M. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, v. 11, p. 93–136, 1996. 14

USCHOLD, M.; JASPER, R. A framework for understanding and classifying ontology applications. In: *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods. Estocolmo, Suécia, ago.,1992*. [S.l.: s.n.], 1992. 14

USCHOLD, M.; KING, M. Towards a methodology for building ontologies. In: *Artificial Intelligence Applications Institute, University of Edinburgh, 1995*. [S.l.: s.n.], 1995. 18

W3C OWL Guide: Site. 2013. Disponível em: <http://www.w3.org/TR/owl-guide/>. Acesso em: 29 ago. 2013.