UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ONTOL-FORUM: LINGUAGEM PARA DESCRIÇÃO DE ONTOLOGIAS APOIADA EM UM MODELO DE REPRESENTAÇÃO DE REGRAS

JÚLIO CÉSAR RODRIGUES DE MENEZES

ORIENTADOR: PROF. DR. LUIZ CAMOLESI JÚNIOR

PIRACICABA, SP

2009

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ONTOL-FORUM: LINGUAGEM PARA DESCRIÇÃO DE ONTOLOGIAS APOIADA EM UM MODELO DE REPRESENTAÇÃO DE REGRAS

JÚLIO CÉSAR RODRIGUES DE MENEZES

ORIENTADOR: PROF. DR. LUIZ CAMOLESI JÚNIOR

Dissertação apresentada ao Mestrado em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como parte dos requisitos para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA, SP

2009

ONTOL-FORUM: LINGUAGEM PARA DESCRIÇÃO DE ONTOLOGIAS APOIADA EM UM MODELO DE REPRESENTAÇÃO DE REGRAS

AUTOR: JÚLIO CÉSAR RODRIGUES DE MENEZES

ORIENTADOR: PROF. DR. LUIZ CAMOLESI JÚNIOR

Dissertação de Mestrado defendida e aprovada em 30 de Junho de 2009, pela Banca Examinadora constituída dos Professores:

Prof. Dr. Luiz Camolesi Júnior – UNICAMP (Orientador)

Prof. Dr. Celmar Guimarães da Silva – UNICAMP

Prof. Dr. Luiz Eduardo Galvão Martins – UNIMEP

À

Minha noiva Michelle pelo apoio e compreensão

À

Minha mãe Selma.

AGRADECIMENTOS

Ao professor Luiz Camolesi Júnior a orientação, compreensão e Incentivo dispensado ao desenvolvimento deste trabalho.

“O Senhor é meu Pastor,

e nada me

faltará.”

Salmo 23:01

ONTOL-FORUM: LINGUAGEM PARA DESCRIÇÃO DE ONTOLOGIAS APOIADA EM UM MODELO DE REPRESENTAÇÃO DE REGRAS

RESUMO

Na busca por uma semântica que modelasse e especificasse dados para a representação do conhecimento, pesquisadores foram incentivados a buscar métodos para esse objetivo. Ontologias foram usadas como um modelo de dados para representar um conjunto de conceitos. A linguagem L-Forum foi criada para especificação das regras modeladas pelo modelo em especificação em regras M-Forum, mas não possuía uma semântica para especificação ontológica de suas dimensões. Este trabalho cria uma extensão à linguagem L-Forum, que se deu pela junção da semântica da linguagem Ontolingua, com a sua própria estrutura, nascendo a linguagem denominada OntoL-Forum, que permite especificar a ontologia das dimensões e associações modeladas pelo modelo M-Forum, melhorando o entendimento das regras envolvidas em um ambiente de interação especificadas pelo modelo.

PALAVRAS-CHAVE: Ontologia, Regra, Forum, OntoL-Forum

ONTOL-FORUM: LANGUAGE TO DESCRIPTION OF THE ONTOLOGY SUPPORTED IN A MODEL OF REPRESENTATION OF RULES

ABSTRACT

In the search for a semantics that modeled and specified data for the representation of the knowledge, researchers were motivated to search methods for this objective. Ontologies were used as a model of data to represent a set of concepts. The language *L-Forum* was created for specification of the rules modeled by the model *M-Forum*, but not possess a semantics for ontological specification of its dimensions. This work creates extension to language *L-Forum*, that felt for the junction of the semantics of the language *Ontolingua*, with its proper structure, being born the called language *OntoL-Forum*, that allows to specify the ontology of the dimensions and associations modeled by the model *M-Forum*, improving the understanding of the rules involved in an environment of interaction specified by the model.

KEYWORDS: Ontology, Rule, Forum, OntoL-Forum

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CSCL – Computer-Supported Collaborative Learning
DL - Description Logics
HTML - Hyper Text Markup Language
KEE - Knowledge Engineering Environment
OWL - Ontology Web Language
RDF - Resource Description Framework
RDF Schema - Resource Description Framework Schema (RDF/S)
RIF - Rule Interchange Format
RIF-BDL - Basic Logic Dialect
UML - Unified Modeling Language
URI - Uniform Resource Identifier
XML - eXtensible Markup Language
W3C - World Wide Web Consortium

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

O computador é uma ferramenta que traz facilidades no mundo moderno, através de programas e aplicativos computacionais processam-se dados transformando-os em informação. Conectado a uma rede, como a *Web*[1], tornou-se uma tecnologia para apresentação e transmissão de dados. As informações da *Web* que outrora eram apresentadas somente para o entendimento humano (BECHHOFER, 2000), podem ser entendidas por máquinas, processadas e apresentadas, devido ao uso de uma semântica que modela e especifica os dados para a representação do conhecimento.

Devido ao acúmulo de informações que são veiculadas pela *Web*, criou-se a *W3C (World Wide Web Consortium)* [2], um consórcio de empresas de tecnologia criado com o objetivo de levar a *Web* ao seu potencial máximo.

A *W3C* propôs a *Web Semântica* como um novo formato de conteúdo para a *Web*, que usa a integração de linguagens e tecnologias, arquiteturas de metadados, agentes computacionais e ontologias, com o objetivo de aumentar a qualidade do resultado das ferramentas de busca através de resolução de ambigüidade e contextualização da informação (SMITH, WELTY & McGUINNESS, 2004).

O uso de ontologias na Ciência da Computação e suas comunidades[3] têm ganhado ênfase como um modelo de dados para representar um conjunto de conceitos dentro de um domínio, como uma forma de representação do conhecimento principalmente na *Web*.

---

[1] *Web* – sigla de *World Wide Web* – em português pode-se entender como: “rede de alcance mundial”

[2] www.w3.org

[3] As comunidades de Inteligência Artificial, Gestão do Conhecimento e a Web.

A linguagem *OWL (Ontology Web Language*)[4], recomendada pela *W3C*, disponibiliza uma forma comum para o processamento de conteúdo semântico da informação na *Web*.

Em um modelo de especificação de regras, o uso de ontologia pode definir e formalizar uma conceitualização dos seres (atores) e objetos envolvidos nas interações dentro de um ambiente (espaço), o tempo e as atividades desenvolvidas, melhorando a legibilidade aos interessados no desenvolvimento de ambientes interativos, facilitando a compreensão e o processamento dos dados envolvidos.

Entre muitos modelos para a especificação de regras, o *M-Forum* (CAMOLESI JR. & MARTINS, 2005) é um modelo de especificação de regras no qual seus desenvolvedores se preocuparam em desenvolver uma semântica que identificasse as dimensões presentes nas interações, apresentando uma formalização para cada dimensão e suas formas de interação. O modelo *M-Forum* possui uma linguagem formal, a *L-Forum*, para a especificação das regras. As regras funcionam como um conjunto de coordenadas para o funcionamento de um determinado sistema para fins de organização.

1.2 OBJETIVO

A ontologia foi introduzida na área da ciência da computação com a finalidade de formalizar conceitos sobre algum domínio que pudesse ser entendido e compartilhado por máquinas e homens (NOY & McGUINNESS, 2001). Já as regras servem para reger as interações, o comportamento e a ordem em que os dados devem ser executados.

Existem várias linguagens para descrição de ontologias, mas a é maioria voltada para *Web*, devido à grande quantidade de informações que são veiculadas pela rede voltada para o interesse de aplicações comerciais, esse é o motivo pelo qual a linguagem *OWL* tem sido uma referência para descrição de ontologias.

A linguagem *Ontolingua* (GRUBER, 1992) foi desenvolvida como um mecanismo para escrita de ontologias, de forma que estas possam ser facilmente traduzidas para uma variedade de representações do conhecimento. Por ser uma linguagem genérica, pode ser facilmente acoplada ao modelo *M-Forum* para a

[4] *OWL (Ontology Web Language)* tornou-se uma recomendação da W3C em 10 de fevereiro de 2004.

especificação de ontologias, criando uma base de conhecimento que representa as dimensões que compõem as regras especificadas pelo *M-Forum*.

O objetivo é estender uma linguagem de especificação de regras para permitir que ontologias possam ser especificadas e usadas na definição de regras. A extensão proposta deve estar alinhada com o modelo formal da linguagem, ou seja, atender as dimensões e interações das regras declaradas em *M-Forum* para se obter um maior entendimento dos seres e objetos envolvidos em um contexto declarado por sua linguagem, a *L-Forum*, criando uma base de conhecimento que possa ser compartilhada e portável para outros sistemas de representação do conhecimento baseado em contexto específico.

1.3 MOTIVAÇÃO

O modelo *M-Forum* em seu projeto inicial foi proposto pelos seus desenvolvedores como um modelo de especificação de regra em ambientes colaborativos (CAMOLESI JR. & MARTINS, 2005), para reger a política de interação neste ambientes.

O homem sempre viveu interagindo com todas as coisas que o cercam, e com a modernidade passou a interagir com máquinas, e por sua vez, a máquina passou a não só interagir com o homem como também com outras máquinas, atores humanos e não-humanos dentro de um contexto.

Com o desenvolvimento e crescimento do modelo *M-Forum*, tornou-se um modelo de especificação de regra capaz de modelar e especificar regras de interação entre atores (seres e objetos) em qualquer ambiente. Sendo assim, foi proposto no Simpósio Brasileiro de Jogos em 2005, como um modelo de especificação de regra em ambiente de jogos, como apóio a projetistas de jogos. Em um ambiente de jogos é possível reconhecer a existência de diversos elementos como: jogadores, objetos envolvidos no jogo, áreas (espaços) de disputa, o tempo do jogo, as atividades que podem ou devem ser realizadas (CAMOLESI JR. & MARTINS, 2005).

A importância da modelagem de regra se deve a necessidade de padronização, clareza, ausência de ambigüidade e consistência, na especificação das interações (CAMOLESI JR. & MARTINS, 2005).

O interesse dos projetistas de jogos em uma linguagem que permitisse a definição dos elementos de um jogo e a especificação de seus inter-relacionamentos impulsionou diversas pesquisas com esse objetivo, propostas envolvendo padrões de projeto, taxonomia de tempo e espaço, representações de ações e eventos (CAMOLESI JR. & MARTINS, 2005).

A motivação para este trabalho é permitir a definição formal dos elementos envolvidos nas interações especificadas através da linguagem *L-Forum*, criando uma estrutura simples que possa ser entendida facilmente e compartilhada. O uso de ontologia pode definir e formalizar uma conceitualização das dimensões e interações envolvidas em um ambiente especificado pela regra, criando uma base de conhecimento.

1.4 METODOLOGIA DO TRABALHO

Para o desenvolvimento do trabalho, foram estudadas linguagens de especificação de ontologias, em especial a *Ontolingua*, suas aplicabilidades e semânticas. O modelo *M-Forum* foi usado para modelar e especificar regras através de sua linguagem formal, a *L-Forum*; exemplos de atividades relacionadas às regras de trânsito vivida no cotidiano exemplificaram a descrição da linguagem.

A semântica da linguagem *Ontolingua*, juntamente com a estrutura da linguagem *L-Forum*, foram integradas para criar uma extensão à linguagem *L-Forum* que permitisse descrever e especificar as ontologias das dimensões e interações das regras declaradas em *M-Forum*.

A extensão recebeu a denominação *OntoL-Forum*, uma linguagem que permite descrever e especificar as ontologias das dimensões e interações das regras declaradas em *M-Forum*. Para exemplificar sua aplicabilidade, foram feitos vários exemplos de estudo de caso, baseados nas regras que exemplificaram o uso da linguagem *L-Forum*.

1.5 ORGANIZAÇÃO DESTA DISSERTAÇÃO

Esta dissertação está dividida em cinco capítulos. No segundo capítulo são tratados os assuntos sobre Ontologia, o que é seus conceitos, aplicação e tipos de linguagem.

No terceiro capítulo são tratados os assuntos sobre o modelo de especificação em regras *M-Forum*, suas dimensões, a linguagem *L-Forum* e suas aplicações.

No quarto capítulo encontra-se a extensão da linguagem *L-Forum*, a definição da linguagem *OntoL-Forum*, as ontologias das dimensões do modelo *M-Forum* em *OntoL-Forum*, sua especificação e aplicação.

O quinto capítulo, por fim, conclui o trabalho, apresentando os resultados e as contribuições da pesquisa e apontando trabalhos futuros.

CAPÍTULO 2

LINGUAGENS PARA ONTOLOGIAS

2.1 CONSIDERAÇÕES INICIAIS

O homem foi impulsionado por suas necessidades a buscar soluções para diversos desafios que os ambientes lhe propuseram, incentivando-o a pesquisa na busca de soluções. Na busca por encontrar uma forma de representação do conhecimento na área computacional, ontologias passam a ser uma ferramenta importante para esse objetivo. Em alguns segmentos da área computacional, como a *Web*, a Inteligência Artificial, entre outros, a maneira encontrada para representar o conhecimento foi através de ontologia. No ambiente computacional o uso de ontologia tem proporcionado, padronização, legibilidade e conhecimento dos seres e objetos dentro do ambiente que a envolve.

2.2 ONTOLOGIA

O termo ontologia é originário da filosofia introduzido por Aristóteles. É um ramo da Filosofia que lida com a natureza e a organização do ser, que busca definir o que é um ser e as características em comum entre outros seres (SEMPREBOM, CAMADA & MENDONÇA, 2007).

Uma ontologia faz uma especificação formal de uma área de conhecimento. “É uma especificação explícita de uma conceitualização. A conceitualização é uma abstração, uma visão simplificada do mundo que nós desejamos representar para alguma finalidade” (GRUBER, 1993).

Foram definidos cinco componentes para a formalização de uma ontologia (GRUBER, 1993):

- Conceitos: podem representar qualquer coisa em um domínio, como uma tarefa, uma função, uma estratégia, etc.
- Relações: representam um tipo de interação entre os conceitos no domínio, sendo a cardinalidade sempre n:n.
- Funções: são casos especiais de relações, sendo a cardinalidade 1:n.

- Axiomas: são sentenças que são sempre verdadeiras, independentemente da situação.
- Instâncias: são utilizadas para representar os elementos do domínio.

Existem diferentes tipos de ontologias de acordo com seu grau de generalidade (GOMES-PEREZ, 1999):

- Ontologias de representação: definem as primitivas de representação – *frames*[5], axiomas, atributos e outros – de forma declarativa;
- Ontologias Gerais: trazem definições abstratas necessárias para a compreensão de aspectos do mundo, como tempo, processos, papéis, espaço, seres, coisas, etc.
- Ontologias Centrais ou Genéricas: definem os ramos de estudos de uma área e/ou conceitos mais genéricos e abstratos desta área. Por exemplo, regras básicas do Direito;
- Ontologias de Domínio: tratam de um domínio mais específico de uma área genérica de conhecimento, como Direito Tributário, Microbiologia, etc.
- Ontologias de Aplicação: procura solucionar um problema específico de um domínio, como identificar doenças do coração, a partir de uma ontologia de domínio de Cardiologia.
- Ontologias de tarefas: descrevem tarefas de um domínio – como processos, planos, metas, escalonamentos etc. – com uma visão mais funcional, embora declarativa, de um domínio;
- Ontologias do domínio: têm uma visão mais epistemológica[6] do domínio, focando nos conceitos e objetos do universo de discurso.

Uma ontologia define um vocabulário de termos comuns para se compartilhar informações sobre um domínio. Isso possibilita criar estruturas de informações e conhecimento compartilhado entre pessoas e agentes de software, podendo ser reutilizado o conhecimento de um domínio, explicitando hipóteses, separando o conhecimento do domínio com o conhecimento operacional e a análise do domínio de conhecimento (NOY & McGUINNESS, 2001).

---

[5] *"Frames"* - Estrutura

[6] Epistemologia ou teoria do conhecimento - é um ramo da Filosofia que trata dos problemas filosóficos relacionados à crença e ao conhecimento. Estuda a origem, a estrutura, os métodos e a validade do conhecimento.

2.2.1 CONCEITOS

O termo Ontologia tem sido adotado pelas comunidades de Inteligência Artificial, Gestão de Conhecimento e pelo *W3C* para se referir a conceitos e termos que podem ser usados para descrever alguma área do conhecimento ou construir uma representação do conhecimento (GUIMARÃES & LUCENA, 2002).

Encontram-se várias definições distintas na literatura para esse termo, pontos de vista distintos e complementares para uma mesma realidade (USEHOLD & GRUNINGER, 1996). Dentro da Ciência da Computação, o significado pode variar conforme o objetivo de uso.

Entre várias definições de ontologia, podem-se destacar as seguintes pela importância:

- É uma especificação formal explícita de uma conceitualização compartilhada (GRUBER, 1993).
- É um conjunto de termos ordenados hierarquicamente para descrever um domínio que pode ser usado como um esqueleto para uma base de conhecimentos (GOMES-PEREZ, 1999).

Na primeira definição, pode-se ressaltar que ontologia se refere a um modelo abstrato de algum fenômeno que identifique conceitos relevantes, devendo ser definidos de forma explícita, trazendo um conhecimento consensual e não restrito a alguns indivíduos. Já na segunda definição, pode-se perceber uma forma complementar da primeira definição, apresentando uma das principais utilidades da ontologia, que é servir como um *"schema"* [7] para uma base de conhecimento, em que uma ontologia deve possuir um conjunto de termos organizados com uma hierarquia associada, ou seja, uma taxonomia (GUIMARÃES & LUCENA, 2002). Assim percebe-se nessa definição algumas informações sobre a estrutura de uma ontologia.

Pode-se também ressaltar na segunda definição a distinção importante entre uma ontologia e uma base de conhecimento. Uma ontologia provê uma estrutura básica na qual se pode construir uma base de conhecimento fornecendo um conjunto de conceitos e termos para descrever um determinado

[7] *Schema* – é uma estrutura conceitual.

domínio. Já a base de conhecimento usa esses termos para descrever uma determinada realidade.

No desenvolvimento de ontologia pode-se considerar alguns passos para desenvolver um projeto de construção (NOY & McGUINNESS, 2001):

- Determinar o domínio e o escopo da ontologia;
- Considerar o reuso de ontologias existentes;
- Enumerar termos importantes da ontologia;
- Definir as classes e suas heranças;
- Definir as propriedades das classes e seus atributos;
- Definir as restrições dos atributos;
- Criar as instâncias.

Um dos benefícios de fator relevante encontrados na utilização de ontologia é o reuso das ontologias e bases de conhecimento pelos desenvolvedores de sistemas baseados em conhecimento (USEHOLD & GRUNINGER, 1996).

2.2.2 APLICAÇÃO

Ontologias são utilizadas em projetos de domínios nas áreas de gestão do conhecimento, comércio eletrônico, processamento de linguagens naturais, recuperação da informação na *Web* (GUIMARÃES & LUCENA, 2002), entre diversas entidades de pesquisas no mundo todo.

Deve-se ressaltar que uma ontologia é um modelo da realidade de um mundo e os seus conceitos devem refletir essa realidade. As classes são o foco de várias ontologias, porque descrevem conceitos de um domínio.

Em termos práticos, o desenvolvimento de ontologias inclui (NOY & McGUINNESS, 2001):

- Definir as classes em uma ontologia;
- Organizar as classes em uma taxonomia hierárquica (subclasse e superclasse);
- Descrever atributos e permitir a descrição de seus valores;
- Preencher os valores de seus atributos para suas instâncias.

Um caso de uso para modelar a aplicação de uma ontologia pode ser visto através da abstração do domínio Veículo apresentado na Tabela 1. O que seria um Veículo? Sabe-se que a entidade Veículo envolve todos os meios para

transportar ou conduzir pessoas ou objetos como: automóveis, motos, caminhões, aviões, etc. A descrição de uma classe Veículo define as entidades Veículos no mundo real.

A especificação da classe Veículo define uma visão geral do domínio através da descrição de suas propriedades.

Tabela 1: Descrição de propriedades de uma classe Veículo

Nome	Descrição da Propriedade
Ano	Ano de fabricação
Combustível	Combustível utilizado
Estado	Novo ou usado
Cor	Cor
Consumo	Quantidade gasta
Fabricante	Quem fabricou
Origem	Local de fabricação

A classe Veículo é uma classe abstrata e não pode ser instanciada, mas serve de modelo para outras classes. A classe Automóvel, por exemplo, é uma subclasse da entidade Veículo, herdando inclusive todas as propriedades desta classe, além das que já possui. Por sua vez, uma entidade automóvel é considerada uma entidade Veículo por estar contida nesse domínio.

2.2.3 LINGUAGENS

Existem várias linguagens para descrição de ontologias, mas a maioria está voltada para *Web*, devido o acúmulo de informações disponíveis que dificultava as pesquisas pela falta de um critério de padronização e organização.

Neste tópico são apresentadas as linguagens *OWL*, *RIF* e *Ontolingua.* Sendo *Ontolingua,* a linguagem utilizada no desenvolvimento deste trabalho.

2.2.3.1 OWL

Na busca por uma padronização e organização da *Web* a *W3C* publicou e recomendou em fevereiro de 1998 a *Extensible Markup Language (XML)*[8] como o novo padrão para representação e troca de dados na *Web* (BRAY, PAOLI & SPERBERG-McQUEEN, 1998). Deste esforço criaram-se sucessivas famílias de

[8] *XML* é uma linguagem extensível; um documento *XML* é formado basicamente por marcações e dados de caracteres.

tecnologias baseadas nesta linguagem, como: *XML Schema*[9], *RDF*[10] e *RDF Schema*[11].

A *OWL* facilita mais a possibilidade de interpretação por máquinas do conteúdo da *Web* do que *XML*, *RDF* e *RDF/S (RDF Schema)*, por fornecer vocabulário adicional com uma semântica formal. A *OWL* é uma revisão da linguagem de *Ontologia Web DAML+OIL* (McGUINNESS & HARMELEN, 2004), que em 2004 passou a ser o padrão recomendado pelo *W3C*.

A *OWL* fornece três sub-linguagens projetadas para o uso de comunidades especificas de implementadores e usuários (SMITH, WELTY & McGUINNESS, 2004):

- *OWL Lite* suporta aqueles usuários que necessitam principalmente de uma classificação hierárquica e restrições simples. Por exemplo, embora suporte restrições de cardinalidade, ela só permite valores de cardinalidade 0 ou 1. É mais simples que as outras sub-linguagens, e faz com que a transição de outros modelos de vocabulários e taxonomias para *OWL* seja mais rápida.
- *OWL DL* suporta aqueles usuários que necessitam de máxima expressividade enquanto mantém a computabilidade (todas as conclusões são garantidas serem computáveis) e decidibilidade (todas as computações terminarão em tempo finito). *OWL DL* inclui todas as construções da linguagem *OWL*, mas impõe algumas restrições (por exemplo, embora uma classe possa ser subclasse de muitas classes (herança múltipla), uma classe não pode ser instância de outra classe). A sigla *DL* é devido a sua correspondência com as lógicas de descrição, um campo de pesquisa que estudou a lógica que forma a base formal da *OWL*.
- *OWL Full* suporta aqueles usuários que necessitam da máxima expressividade e a liberdade sintática do *RDF* sem nenhuma garantia computacional. Por exemplo, em *OWL Full* uma classe pode ser tratada simultaneamente como uma coleção de indivíduos e como um indivíduo por si mesma. *OWL Full* permite que uma ontologia aumente o vocabulário pré-

[9] *XML Schema* fornece um meio de especificar que elementos e atributos podem ser incluídos em um documento *XML*, ou seja, possibilita a validação do documento.
[10] *RDF* é um modelo para representação de dados na *Web*.
[11] *RDF Schema* é um mecanismo para representação de dados. É utilizado para estruturar a informação contida nos modelos *RDF*.

definido de *RDF* ou *OWL*. É improvável que algum software de inferência venha a ser capaz de suportar completamente cada recurso da *OWL Full*.

Todas as sub-linguagens de *OWL* fazem uso da sintaxe *RDF* na construção de ontologias. De acordo com a *W3C* a *Ontology Web Language OWL* foi projetada de acordo com os seguintes objetivos (ZHIHONG e MINGTIAN, 2003):

- Ontologias Compartilhadas: As ontologias devem ser acessadas publicamente e de diferentes fontes web, permitindo que a mesma ontologia seja compartilhada.
- Evolução da Ontologia: Uma ontologia pode sofrer mudanças ou ser atualizada durante o seu tempo de vida. A fonte web deve especificar a versão da ontologia.
- Interoperabilidade de Ontologia: Diferentes ontologias podem modelar o mesmo conceito em diferentes maneiras. A linguagem deve fornecer base para relatar diferentes representações, assim permitindo que os dados sejam convertidos para diferentes ontologias e permitindo uma “web de ontologias”.
- Detecção de Inconsistência: Diferentes ontologias ou fontes de dados podem ser contraditórias. Deve ser possível detectar essas inconsistências.
- Um balanço entre a expressividade e a dimensão: A fim de expressar uma grande variedade de conhecimento, a linguagem deve ter um poder expressivo forte, enquanto também deve fornecer meios eficientes para um raciocínio avançado. Estas duas exigências estão geralmente em desacordo, pois o objetivo da linguagem para ontologia web é encontrar um equilíbrio que suporte a capacidade de expressar a maioria dos tipos importantes de conhecimento.
- Facilidade de Uso: A linguagem deve fornecer uma baixa barreira de aprendizado e ter conceitos e significados claros. Os conceitos devem ser independentes da sintaxe.
- Compatibilidade com outros padrões: A linguagem deve ser compatível com outro padrão web e padrões comerciais de uso geral. Em particular,

isto inclui *XML* e padrões relacionados (tais como *XML Schema* e *RDF/S*), e possivelmente outros padrões de modelagem, como a *UML*[12].

- Internacionalização: A linguagem deve dar suporte para o desenvolvimento de ontologias em diversas línguas, e fornecer possibilidades diferentes de visão de ontologias, apropriadas para diferentes culturas.

Uma ontologia *OWL* descreve a estrutura de um domínio em termos de classes e propriedades de forma parecida com a orientação a objetos. Uma ontologia é feita de um conjunto de axiomas declarados supondo o relacionamento entre as classes e as restrições de suas propriedades.

A *OWL* possui uma sintaxe abstrata que é uma seqüência de axiomas e fatos que inclui referências para outras ontologias. As ontologias *OWL* são documentos *Web* que podem ser referenciados através de *URI*[13].

A Figura 1 mostra a sintaxe abstrata da *OWL* que descreve a estrutura das ontologias:

```
<ontology> ::= Ontology ( {<directive>} )
<directive> ::= Annotation ( <URI ref><URI ref> )
<directive> ::= Annotation ( <URI ref><lexical-form> )
<directive> ::= Imports ( <URI> )
<directive> ::= <axiom>
<directive> ::= <fact>
```

Figura 1 – Sintaxe Estrutura da Ontologia em *OWL* (ZHIHONG e MINGTIAN, 2003).

Existem dois tipos de fatos na abstração da sintaxe *OWL*. O primeiro tipo é a informação do estado de um indivíduo em particular, sob a forma das classes a que o indivíduo pertence, além dos valores e propriedades individuais. Para um indivíduo pode ser dado um indiID, Figura 2, identificador que denota o indivíduo e que pode ser usado para referenciá-lo. Entretanto, indivíduos não precisam de um indiID; neste caso, esses indivíduos são chamados de anônimos, ou seja, em branco em termos de *RDF,* não podendo ser diretamente referenciados em outra parte (ZHIHONG e MINGTIAN, 2003).

---

[12] *UML* - em português significa Linguagem de Modelagem Unificada, padrão OMG (Object Management Group) para o desenvolvimento de sistemas baseados na orientação a objetos.

[13] *URI* - em português significa Identificador Uniforme de Recursos, é responsável pela codificação de cada um dos endereços da Web: um documento HTML, imagem, videoclipe, programa etc.

```
<fact> ::= <indi>
<indi> ::= individual ( [<indiID>] {<annotation>}
              {type (<type>)} {<proValue>} )
<proValue> ::=  value (<indi_vProID>< indiID >)
                  | value (<indi_vProID>< indi>)
                  | value (<data_vProID>< dataLiteral >)
```

Figura 2 – Sintaxe para Indivíduos em *OWL* (ZHIHONG e MINGTIAN, 2003).

Na classe *OWL,* da Figura 3, os axiomas são usados para estipular que uma classe é exatamente equivalente, para a modalidade completa, ou uma subclasse, para a modalidade parcial, à conjunção de uma coleção de superclasses e restrições *OWL* (ZHIHONG e MINGTIAN, 2003).

```
<axiom> ::= Class ( <classID> <madality>
              { <annotation> } { <description>} )
<modality> ::= complete I partial
<axiom> ::= disjointWith (<description> { <description> } )
<axiom> ::= sameClassAs ( <description> { <description>} )
<axiom> ::= subClassOf ( <description> <description>)
```

Figura 3 – Sintaxe para Axioma em *OWL* (ZHIHONG e MINGTIAN, 2003).

Descrições em uma sintaxe abstrata completa incluem ID´s de classe e construtores de restrições. As descrições da Figura 4 podem ser também combinações booleanas de outras descrições, e conjuntos de indivíduos (ZHIHONG e MINGTIAN, 2003).

```
<description> ::= <classID>
                  I <restriction>
                  I unionOf ( { <description> } )
                  I intersectionOf ( { <description> } )
                  I complementOf ( <description> )
                  I one of ( { <indiID> } )
```

Figura 4 – Sintaxe para Descrições em *OWL* (ZHIHONG e MINGTIAN, 2003).

Na sintaxe abstrata *OWL*, as propriedades da classe podem ter valores determinados. Como também, as cardinalidades da Figura 5 podem ser um conjunto de alguns números (cardinalidade mínima ou máxima) (ZHIHONG e MINGTIAN, 2003).

```
<restriction> ::= restriction ( data_vProID>
                  { allValuesFrom ( <dataRange> ) }
                  { someValuesFrom ( <dataRange> ) }
                  {value (<dataLiteral> ) } { <cardilaty> } )
<restriction> ::= restriction ( indi_vProID>
                  { allValuesFrom ( < description> ) }
                  { someValuesFrom ( <description> ) }
                  { value ( < indiID> ) } { <cardinality> } )
<cardinality>::= minCardinality ( <non-negative-integer> )
                  I maxCardinality ( <non-negative- integer>)
                  I Cardinality ( <non-negative-integer> )
```

Figura 5 – Sintaxe para Restrições *OWL* (ZHIHONG e MINGTIAN, 2003).

A figura 6 mostra que os axiomas de propriedade permitem descrições para si mesmo, nas classes e no campo de dados dos tipos de dados, domains e ranges.

```
<axiom> ::= DatatypeProperty ( <data_vProID>
            {<annotation>} {super ( <data_vProID>)}
            {domain (<description>)}
            {range (<dataRange>) } {Functional})
<axiom> ::= ObjectProperty ( <indi_vProID>
            {<annotation>} {super (<indi_vProID>)}
            {domain (<description>)}
            {range (<description>)}
            [inverseOf (<indi_vProID>)] [Symmetric]
            [Functional | InverseFunctional
            | Transitive ])
<axiom> ::= samePropertyAs (<data_vProID>
            {<data_vProID>})
<axiom> ::= subPropertyOf (<data-vProID>
            <data_vProID>)
<axiom> ::= samePropertyAs (<indi_vProID>
            {<indi_vProID>})
<axiom> ::= subPropertyOF (<indi_vProID>
            <indi_vProID>).
```

Figura 6 – Sintaxe para Axioma de Propriedade em *OWL* (ZHIHONG e MINGTIAN, 2003).

Na declaração de um documento *OWL* pode-se incluir um cabeçalho "*ontology*" opcional e nenhum número de classes, propriedades, declaração de indivíduos ou axiomas (DING & PENG, 2004). Segue um resumo de um conjunto de construtores usado na linguagem *OWL* para formar as declarações ou axiomas.

Uma classe define um grupo de indivíduos que compartilham algumas propriedades. Cada indivíduo na *OWL* é membro da classe *owl:Thing*, sendo ela superclasse de todas as classes *OWL* definidas pelos usuários.

Uma classe é sintaticamente representada como uma instância nomeada da *owl:Class* e pode ser definida da seguinte maneira: *"<owl:Class*

rdf:ID=”Animal” />” que define uma classe “Animal” que é uma instância de “owl:Class”. Uma classe anônima pode ser declarada para exaustivamente enumerar todos os indivíduos que são instâncias desta classe *(owl:oneOf),* para uma restrição de propriedade *(owl:Restriction)*, ou para operação lógica de duas ou mais classes *(owl:intersectionOf, owl:unionOf, owl:complementOf)*. As restições de propriedades incluem valores *(owl:allValuesFrom, owl:someValuesFrom, owl:hasValue)* e restrições de cardinalidade *(owl:cardinality, owl:maxCardinality, owl:minCardinality)* (DING & PENG, 2004). A Figura 7 mostra um valor de restrição:

```
<owl:Restriction>
    <owl:onProperty rdf:resource=”#hasParent”/>
    <owl:allValuesFrom rdf:resource=”#Human”/>
</owl:Restriction>
```

Figura 7 – Restrição em *OWL* (DING & PENG, 2004).

Foi definida uma classe anônima de todos os indivíduos cuja propriedade ”hasParent” somente tem valores pertencente a classe “Human”. A Figura 8 mostra uma restrição de cardinalidade que define uma classe de todos os indivíduos que tem somente um “PAI”.

```
<owl:Restriction>
    <owl:onProperty rdf:resource=”#hasFather”/>
    <owl:cardinality rdf:datatype=”&xsd;nonNegativeInteger”>1
    </owl:cardinality>
</owl:Restriction>
```

Figura 8 – Restrição de Cardinalidade em *OWL* (DING & PENG, 2004).

Existem três operadores lógicos que correspondem ao *AND (conjunção), OR (disjunção)* e *NOT (negação)* lógicos, e que definem a classe de todos os indivíduos para as operações de conjunto padrão: união, intersecção e complemento. A *OWL* possui três axiomas de classe *(rdfs:subClassOf, owl:equivalentClass, owl:disjointWith)* .

```
<owl:Class rdf:ID=”Female”>
    <rdfs:subClassOf rdf:resource=”#Animal”/>
    <owl:disjointWith rdf:resource=”#Male”/>
</owl:Class>
```

Figura 9 – Operações de Conjunto em *OWL* (DING & PENG, 2004).

O exemplo da Figura 9 define que a classe “Female” é uma subclasse da classe “Animal” e é disjunta da classe “Male”, ou seja, as duas classes não têm indivíduos em comum. Mais dois tipos de propriedades podem ser definidos em *OWL*: propriedade de objeto *(owl:ObjectProperty)* que liga os indivíduos uns aos outros, e propriedade dos tipos de dados *(owl:DatatypeProperty)* que liga os indivíduos aos valores dados. Similarmente para as classes, ”rdfs:subPropertyOf” é usado para definir que uma propriedade é uma sub-propriedade de outra propriedade.

O exemplo da Figura 10 mostra que “hasParent” é uma propriedade de objeto que tem como domínio a classe “Animal”, isso significa que, esta propriedade “hasParent” deve ser usada somente por indivíduos da classe “Animal”, e o valor *rdfs:range*[14] é da classe “Animal”, isso significa que os valores desta propriedade devem pertencer à extensão da classe “Animal”.

```
<owl:ObjectProperty rdf:ID=”hasParent”>
    <rdfs:domain rdf:resource=”#Animal”/>
    <rdfs:range rdf:resource=”#Animal”/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID=”hasFather”>
    <rdfs:subPropertyOf rdf:resource=”#hasParent”/>
    <rdfs:range rdf:resource=”#Male”/>
</owl:ObjectProperty>
```

Figura 10 – Restrição de Propriedade em *OWL* (DING & PENG, 2004).

Já a propriedade de objeto “hasFather” é definida como uma sub-propriedade de “hasParent”, e herda o mesmo domínio de “hasParent”, e o valor *rdfs:range* deve ser de indivíduos da classe “#Male”.

Existem mais outros construtores *OWL* que são usados conforme a abstração que os envolve. A semântica *OWL* é definida de maneira análoga à semântica da *Description Logics (DL)*[15].

[14] *rdfs:range* - indica a que classe deve pertencer o objeto de uma propriedade, é uma restrição herdada do *RDF Schema*.

[15] *Description Logics* – em português significa Lógicas de Descrição. É uma área de pesquisa que estuda um fragmento particular da lógica de primeira ordem.

2.2.3.2 RULE INTERCHANGE FORMAT (RIF)

O *Rule Interchange Format (RIF)* é um formato recomendado pelo *W3C* para fornecer suporte ao intercâmbio das diversas tecnologias baseadas em regras (BRUIJN, 2008). Um esforço de um grupo de trabalho no desenvolvimento de um formato para troca de informações entre várias linguagens de regras para serem usadas por diferentes sistemas na *Web Semântica* (BOLEY et al., 2007). A iniciativa do *RIF* é estreitamente relacionada às Ontologias.

O objetivo fundamental de *RIF* é ser um meio eficaz de troca de informação entre regras, facilitando o intercâmbio das regras e permitindo o compartilhamento da informação em uma forma adequada para o processamento de máquina e consistente com as tecnologias existentes e as especificações do *W3C* (PASCHKE et al., 2008).

As regras que fazem intercâmbio usando *RIF* podem referenciar fontes de dados externas e podem ser baseadas em modelos de dados (ontologias) que são representados através de uma linguagem diferente (BRUIJN, 2008).

O *RIF* fornece múltiplas versões de linguagens para especificação de regras no intercâmbio de informações, chamados de dialetos. Os dialetos *RIF* foram elaborados primeiramente para intercâmbio, entretanto, cada dialeto é uma linguagem de regra padrão e pode ser usado mesmo quando não são requeridos para portabilidade e intercâmbio. (RIF WORKING GROUP, 2008)

Os dialetos padrões de RIF são (RIF WORKING GROUP, 2008):

- *CORE* – Este é a linguagem básica de *RIF*. Ele é elaborado para ser um subconjunto comum de várias regras de máquinas.
- *BDL* (*Basic Logic Dialect)* – Este acrescenta algumas coisas que não se encontra no *CORE*: funções lógicas, igualdade no *"then-part*[16]*"* e argumentos nomeados.
- *PRD (Production Rules Dialect)* – Este acrescenta uma noção de regras *"forward-chaining*[17]*"*, onde uma regra dispara e executa alguma ação, tal como acrescenta mais informações para armazenar ou obtém alguma informação.

[16] O termo *"then-part*" refere-se à uma segunda parte que pode ser desenvolvida numa expressão lógica.
[17] Forward-chaining é um método usado para fazer inferência até que o objetivo seja alcançado.

O *RIF-BDL (Basic Logic Dialect)* é o formato usado para intercâmbio de regras lógicas sobre a *Web* (BRUIJN, 2008), suportando as linguagens *RDF*, que é uma linguagem para representação e intercâmbio de dados, e as linguagens *RDF Schema* e *OWL* que são linguagens para representação e intercâmbio de ontologias.

A linguagem *OWL*, é uma extensão de suas antecessoras *XML Schema*, *RDF* e *RDF Schema*, ou seja, a combinação de uma linguagem *OWL* e o *RIF-BDL*, também é uma extensão da combinação da linguagem *RDF* com o *RIF-BDL* (BRUIJN, 2008).

A sintaxe para intercâmbio de ontologias *OWL* são baseados em grafos *RDF* (*RDF* é um modelo que usa uma sintaxe de especificação baseada *XML*). Entretanto, a combinação *RIF-OWL* consiste de um documento *RIF* e um conjunto de grafos *RDF* (BRUIJN, 2008).

Existe uma declaração correspondente entre grafos *RDF* e certos tipos de documentos *RIF*. Esta é uma correspondência entre triplas *RDF* de forma *s p o*[18] e o formulário de estrutura *RIF* da forma s'[p' -> o'], onde *s', p'* e *o'* são símbolos *RIF* correspondentes para os símbolos *s, p,* e *o* de *RDF*. Sempre que uma tripla *RDF* é satisfeita a correspondente *RIF* também é satisfeita, ou vice-versa (BRUIJN, 2008).

Considerando a Figura 11, por exemplo, que mostra a combinação de um grafo *RDF* que contém uma tripla dizendo que, ***ex: john*** é irmão de ***ex: jack*** é parente de ***ex: mary***, e um documento *RIF* que diz que quando algum **x** é irmão de algum **y** e **y** é parente de algum **z**, então **x** é algum tio de **z** (BRUIJN, 2008).

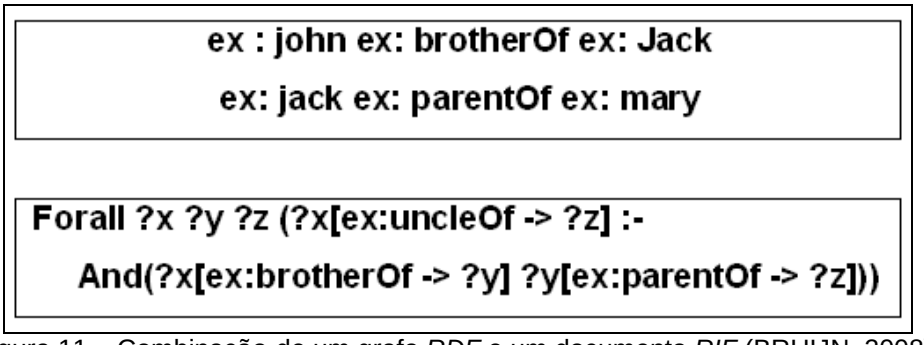

Figura 11 – Combinação de um grafo *RDF* e um documento *RIF* (BRUIJN, 2008).

[18] Uma declaração *RDF* é formada de três partes: sujeito, predicado e objeto.

Para esta combinação da fórmula de estrutura *RIF* ***:John [:uncleOf -> :mary]***, assim como a tripla *RDF* ***:john :uncleOf :mary***, pode ser derivado (BRUIJN, 2008).

A semântica das combinações é definida nos termos dos modelos combinados, que são pares de interpretações de *RIF* e de *RDF*. A interação entre as duas interpretações é definida com um número de circunstâncias.

2.2.3.3 ONTOLINGUA

A *Ontolingua* (GRUBER, 1992) foi introduzida inicialmente em 1990 por Thomas R. Gruber, e concluída em 1992. É um mecanismo para escrita de ontologias, de forma que estas possam ser facilmente traduzidas para uma variedade de representações. A definição de sua sintaxe e semântica é baseada em uma versão estendida do cálculo de predicados de primeira-ordem conhecida como *Knowledge Interchange Format (KIF)*. O *KIF* foi estendido como uma linguagem para a comunicação e publicação do conhecimento, com objetivo de deixar o conteúdo de uma base de conhecimento claro para o leitor, expressivo, mas não dava suporte para o processamento automático nesta forma (GRUBER, 1993).

Uma notação *Lisp-Like*[19] é fornecida em *KIF* para declarar os axiomas definidos na *Ontolingua*, sintaxe prefixa para cálculo de predicados com termos funcionais e equilibrados. Os objetos são denotados por objetos constantes ou expressões de termos que são construídos de uma lista onde o primeiro elemento é uma função constante. As sentenças são formadas de uma lista onde o primeiro elemento é uma relação constante e os elementos restantes são termos, ou por operações lógicas sobre tais sentenças (GRUBER, 1993).

As variáveis podem ser determinadas universalmente ou existencialmente por uma marcação prefixa ‘?’. Existem também operadores para a junção, a disjunção, a implicação, e a negação. Para exemplificar, a sentença da Figura 12 em KIF diz: “todos os escritores são mal entendido por alguns leitores” (GRUBER, 1992).

[19] Lisp é uma linguagem de programação criada em 1958 por John McCarthy. É uma linguagem de expressão orientada. Ao contrário da maioria das outras linguagens, não faz distinção entre a expressão e declaração, todo o código e dados são declarados como expressões.

```
(forall ?W
        (=> (writer ?W)
                (exists (?R ?D)
                        (and (reader ?R)
                        (document ?D)
                        (writes ?W ?D)
                        (reads ?R ?D)
                        (not (understands ?R ?D))))
```

Figura 12 – Exemplo de sentença em KIF (GRUBER, 1992).

- O símbolo *writer* é uma relação constante e a sentença *(writer ?W)* diz que *?W* é um escritor, e o '=>' é uma implicação. O símbolo *?W* é uma variável determinada universalmente, e *?R* e *?D* são existencialmente determinadas.
- O símbolo *writes* é relação binária, e a sentença *(writes ?W ?D)* diz que ?W escreve o documento ?D.
- O símbolo *reads* é relação binária, e a sentença *(reads ?R ?D)* diz que ?R lê o documento ?D.
- O símbolo *understands* é relação binária, e a sentença *(understands ?R ?D)* diz que ?R entende o documento ?D. Mas o símbolo *not* faz a negação da expressão anterior.
- As variáveis determinadas *?W* e *?D* não são tipificadas ou classificadas, embora na prática os tipos são restritos com um predicado unário, tais como *writer* e *document*.

A *Ontolingua* traduz as definições escritas em *KIF* introduzindo uma forma apropriada para implementar sistemas de representação, que impõe uma sintaxe restrita e suporta argumento limitado sobre um subconjunto de restrição completa para uma lógica em primeira ordem.

As definições em *Ontolingua* são na forma *Lisp-style* que associa um símbolo a uma lista de argumentos, uma string de declaração e um conjunto de sentenças *KIF* por palavras chaves.

Esses são os requisitos do projeto *Ontolingua* (GRUBER, 1992):

- Possuir uma boa definição, semânticas declarativas para todas as declarações na linguagem.

- Possuir um mecanismo que permite uso operacional das ontologias em uma variedade de sistemas de representação implementados.
- Possuir uma sintaxe que facilite a definição de parte dos termos em uma ontologia, e partes de pacotes de ontologias.
- Meios de capturar conversões em uma representação e organização do conhecimento, cada classe com sua hierarquia e domínio, restrições e relacionamentos. Em um sistema independente, formas declarativas que valorize a implementação eficiente destas conversões para vários sistemas de representação.
- Possuir uma arquitetura e biblioteca para suportar e fazer escrita adicional em uma tradução KIF.

2.2.3.3.1 DEFINIÇÕES EM ONTOLINGUA

A *Ontolingua* consiste de uma forma para definição de classes, relacionamentos, funções, objetos, teorias e afirmação de relatos feita sobre elas. Uma sintaxe de definição em *Ontolingua* é constituída por: um nome, uma lista de argumentos, string de declarações, e um conjunto de campo de sentenças *KIF* (GRUBER, 1992).

A Figura 13 mostra a forma usada em *Ontolingua* para definir uma classe.

```
1 (define class class_name (? instance-variable))
2       "documentação string"
3       :def or :iff-def KIF sentence
4       :constraints KIF sentence
5       :sufficient KIF sentence
6       :equivalent KIF sentence
7       :default-constraints KIF sentence
8       :axioms KIF sentence )
```

Figura 13 - Uma sintaxe de definição em *Ontolingua* (GRUBER, 1992).

Na linha 1, se define o nome da classe, e o termo (*? Instance-variable)*, em uma lista de argumento, é uma variável *KIF* determinada universalmente para as instâncias das classes. Na linha 2, o termo *"documentação string"* descreve algo sobre a classe. Na linha 3, a sentença do campo *:def*

especifica as condições necessárias de associação das classes, e a sentença do campo *:iff-def* são especificações necessárias e as condições suficientes para a associação. Nas linhas 4 e 6, as sentenças dos campos *:constraints* e *:equivalent* também especificam as condições necessárias e as condições necessárias – e – suficientes, respectivamente. Elas são restrições sobre as instâncias da classe, mas não são consideradas intrínsecas para a definição do termo. Na linha 5, a sentença do campo *:sufficient* descreve uma suficiência, uma condição que não é necessária, apenas suficiente. Na linha 8, a sentença do campo *:axioms* é uma sentença que usa os termos inicialmente definidos, como o nome da classe, mas não faz menção à variável instanciada. Esta permite escrever um axioma sobre a classe como um objeto, ou expressões complexas sobre as instâncias da classe qual não cabe no formato das condições necessárias e suficientes.

A sintaxe e a semântica da linguagem *Ontolingua* são escritas em linguagem natural, cujo texto não é analisado gramaticalmente ou traduzido, passado para outro sistema com um lugar específico para documentação. A tarefa é traduzir em sentenças declarativas, axiomas que tem um significado de termos definidos.

Em *Ontolingua* as classes, relações e funções são definidas com a mesma sintaxe, apoiada pelos mesmos campos de sentenças. A diferença é que na definição das relações e funções uma lista de um ou mais argumentos é permitido, já as classes têm uma única instância de variável. Também as classes, relações e funções são definidas como conjunto de tuplas, uma sentença da forma (*predicado* $arg_1 ... arg_n$), significando que a relação denotada pelo predicado se estende a todos os argumentos. Se houver a existência de um único argumento, significa que arg_1 é uma instância de uma classe denotada por predicado.

2.2.3.3.2 CLASSE

Uma classe pode ser definida como uma coleção de indivíduos que são instâncias da Classe. Em *Ontolingua* uma Classe é uma relação unária que possui somente um argumento, que é a variável instanciada. Em *Ontolingua*, no *Frame Ontology,* o conceito de classe é definido como a classe das classes como mostra a Figura 14.

```
(define-class CLASS (?class)
  "A class is a unary relation."
  :iff-def (and (relation ?class)
                (= (arity ?class) 1)))
```

Figura 14 – Modelo conceitual de uma Classe (GRUBER, 1992).

Muitos dos termos definidos em ontologias são classes. Aqui a definição de uma classe é um predicado sobre uma variável livre, onde o predicado se mantém sobre todas as instâncias da classe.

Considerando como exemplo uma ontologia para informação de bibliografia, que inclui documentos, autores, publicações, datas, lugares e referência que ocorre em documentos e catálogos de cartão. Pode-se ter o seguinte exemplo na Figura 15:

```
(define-class AUTHOR (?author)
  "An author is a person who writes things. An author must have created at least one document.
  In this ontology, an author is known by his or her real name."
    :def (and (person ?author)
              (= (value-cardinality ?author AUTHOR.NAME) 1)
              (value-type ?author AUTHOR.NAME biblio-name)
              (>= (value-cardinality ?author AUTHOR.DOCUMENTS) 1)
              (<=> (author.name ?author ?name)
                   (person.name ?author ?name))))
```

Figura 15 – Exemplo para definir uma classe Autor (GRUBER, 1993).

Esta forma define o termo autor, que denota a classe. Autores individuais são instâncias dessa classe.

- A sentença ***:dep*** tem as condições necessárias para associação na classe. Já o primeiro conjunto ***(person ?author)***, descreve que todo autor deve ser uma pessoa.
- O segundo conjunto ***(= (value-cardinality ?author AUTHOR.NAME) 1)***, descreve que autores devem ter exatamente um nome associado, dado pela relação *author.name*. Isto significa que cada instância da classe está ligada a somente um nome de autor.

- O terceiro conjunto ***(value-type ?author AUTHOR.NAME blibio-name)***, especifica o tipo de restrição para o valor da relação *author.name* aplicada para as instâncias da classe author, que deve ser uma instância da classe *biblio-name*.
- O quarto conjunto ***(>= (value-cardinality ?author AUTHOR.DOCUMENTS) 1)***, descreve que alguns documentos devem ser associados através da relação *author.documents*, e mostra que todo autor tem pelo menos um trabalho publicado. Já o quinto conjunto especifica que ***author.name*** e ***person.name*** são as mesmas pessoas.

2.2.3.3.3 RELAÇÃO

Uma relação é um conjunto de tuplas que representa um relacionamento entre os objetos em um universo de discurso. Cada tupla é uma seqüência finita ordenada de objetos.

Em KIF as seqüências são chamadas de listas, não como estruturas de dados, mas como entidades abstratas. As relações são denotadas por predicados. Em *Ontolingua*, a maneira de conceitualização de relações é capiturada na seguinte definição, como mostra a Figura 16.

```
(define-class RELATION (?relation)
  "A relation is a set of tuples."
  :iff-def (and (set ?relation)       ; ?relation is a set
                (=> (member ?tuple ?relation)
                    (list ?tuple))))  ; ?tuple is a sequence
```

Figura 16 – Modelo conceitual de uma Relação (GRUBER, 1992).

O conceito de relação, definido na Figura 16, mostra que uma relação possui um nome e é instanciada por uma variável com uma marcação prefixa ‘?’. Depois, pode-se ser visto uma declaração conceitual sobre a relação, que neste caso, diz que uma relação é um conjunto de tuplas. No campo ***:iff-def*** são feitas as especificações necessárias e as condições suficientes. Dentro deste campo, a variável instanciada é confirmada como um conjunto, sendo parte de uma tupla, definida como uma lista.

Podemos classificar uma tupla em particular como um elemento de uma relação denotada por (nome-relação arg_1 arg_2... arg_n), onde o arg_1 é o objeto na tupla. No caso de relação binária pode-se ser lido como arg_1 nome-relação arg_2.

Por exemplo, uma relação “Connects” pode ser definida através da Figura 17.

```
(define-relation CONNECTS (?comp1 ?comp2)
   "The most general binary connection relation between components.
   Connected components cannot be subparts of each other."
   :def (and (component ?comp1)
             (component ?comp2)
             (not (subpart-of ?comp1 ?comp2))
             (not (subpart-of ?comp2 ?comp1)))
```

Figura 17 – Exemplo para definir uma relação Connects (GRUBER, 1993).

Os argumentos ***?comp1*** e ***?comp2***, mostrado na Figura 17, são variáveis universalmente determinadas, agrupadas sobre os itens nas tuplas da relação. Este exemplo é uma relação binária, cada tupla na relação tem dois itens, podendo também ser definido um número maior de argumentos.

A sentença que vem depois da palavra chave ***:def*** é uma sentença *KIF* que indica uma restrição lógica sobre os argumentos. As restrições no valor do primeiro argumento de uma relação binária são efetivamente restrição de domínio, e esses no segundo argumento são restrições de valor. Podendo também ser expressões complexas indicando relacionamento entre os argumentos para uma relação. As restrições no campo ***:def*** são condições necessárias, as quais devem ser mantidas se a relação as mantêm sobre alguns argumentos.

2.2.3.3.4 FUNÇÃO

Uma função é uma relação especial. Conceitualmente, as funções são conjuntos de tuplas apenas como relações ordenadas, em que o último item da *n-ésima* tupla é o valor da função no primeiro item *n-1*da tupla.

Uma função unária é uma relação binária. Formalmente, uma função é definida como mostra a Figura18.

```
(define-class FUNCTION (?relation)
  "A function is a mapping from a domain to a range that associates a domain element with
exactly one range element."
  :iff-def (and (relation ?relation)
                (=> (member ?tuple1 ?relation)
                    (member ?tuple2 ?relation)
                    (= (but-last ?tuple1) (but-last ?tuple2))
                    (= (last ?tuple1) (last ?tuple2)))))
```

Figura 18 – Modelo conceitual de uma Função (GRUBER, 1992).

Uma função é definida com a mesma semântica da relação, com uma leve variação em sua sintaxe no final do argumento fora da lista. Exemplo, Figura 19.

```
(define-function SQUARED (?n) :-> ?value
   "The squared of a number is the product of it times itself."
   :def (and (number ?n)
             (nonnegative-number ?value))
   :lambda-body (* ?n ?n))
```

Figura 19 – Exemplo para definir uma função Squared (GRUBER, 1993).

Na definição das relações os argumentos para uma função são restringidos como condições necessárias depois da palavra chave ***:def*** . A função é definida somente em argumentos que satisfazem essas restrições de domínio. No caso da Figura 19, a condição ***(number ?n)***, significa que a função é definida somente por números.

Uma restrição no valor da função é especificada com uma sentença que restringe o valor da variável após a palavra-chave **:->**, ou seja, a variável ***?value*** é um número não negativo.

A forma de definição do corpo da função e da relação é semelhante, mas encontramos na função uma sentença ***:lambda-body*** que é um termo de expressão *KIF* que denota o valor da função e de seus argumentos. No exemplo da Figura 19, o termo ***:lambda-body*** denota o produto da variável ***?n*** por ela mesma, onde o ***** é uma função.

2.3 CONSIDERAÇÕES FINAIS

O conteúdo aqui abordado traz o conhecimento fundamental para a realização deste trabalho. Pôde ser visto a definição de Ontologia, conceitos, os componentes para sua formalização, os tipos de ontologias e algumas linguagens que especifica ontologias. As linguagens apresentadas foram: *OWL*, *RIF* e *Ontolingua*. A linguagem *OWL* mostra uma estrutura bem definida, uma semântica consistente, mas foi elaborada exclusivamente para formalização do conteúdo semântico disponibilizado na *Web*, o motivo que a tornou importante na especificação de ontologias. A linguagem *RIF* é usada para troca de informações entre várias linguagens de regras, pode referenciar fontes de dados externas e pode ser baseadas em modelos de dados, mas seu uso também é direcionado para o conteúdo *Web*.

A *Ontolingua* é uma linguagem mais genérica e de fácil entendimento, sua sintaxe e semântica são escrita em linguagem natural e as especificações de classes, relações e funções, são apoiadas pelos mesmos campos de sentenças. Por esse motivo a linguagem *Ontolingua* mostrou-se a mais adequada a ser usada no desenvolvimento deste trabalho.

CAPÍTULO 3

MODELO DE REGRAS M-FORUM

Criado pelo Grupo de pesquisas em Ambientes Colaborativos (GAC - FACEN/UNIMEP), o *M-Forum* é um modelo de regras que teve sua primeira versão finalizada em 2004 ao contemplar conceitos elementares da especificação de regras baseadas em eventos (BRESSAN, 2008).

Os conceitos do modelo são definidos através da teoria de conjuntos e as regras são definidas por uma linguagem formal, a *L-Forum*. Este modelo e sua linguagem foram desenvolvidos para a especificação de regras de interações em ambientes colaborativos. Os elementos envolvidos nas regras são denominados *dimensões*, as quais são: atores, atividades, objetos, tempo, espaço e as associações entre elas (CAMOLESI JR. & MARTINS, 2005).

3.1 DIMENSÕES

As dimensões são elementos envolvidos nas interações entre seres (atores) e objetos dentro de um ambiente (espaço) formalizado por um período de tempo em que as atividades devem ser desenvolvidas.

No modelo *M-Forum* as cinco dimensões relacionadas em uma interação são representadas por atores *(Ac_m)*, atividades *(At_m)*, objetos *(Ob_m)*, tempo *(Ti_m)* e espaço *(Sp_m)*, com o envolvimento de operadores para as respectivas dimensões *($atop_m$, $spop_m$ e $tiop_m$)* para serem reconhecidos e modelados (CAMOLESI JR. & MARTINS, 2005).

$$InteracSS = AcSS \times Ao \times AtSS \times ObSS \times So \times SpSS \times To \times TiSS$$
$$Interac\_Sentence_m \in InteractionSS$$
$$Interac\_Sentence_m = (Ac_m, atop_m, At_m, Ob_m, spop_m, Sp_m, tiop_m, Ti_m)$$
$$Ac_m \in AcSS,\ At_m \in AtSS,\ Ob_m \in ObSS,\ Sp_m \in SpSS,\ Ti_m \in TiSS$$
$$atop_m \in Ao,\ spop_m \in So,\ tiop_m \in To$$

Figura 20 – Dimensões representadas por teoria de Conjuntos (CAMOLESI JR. & MARTINS, 2005).

A Figura 20 mostra uma visão geral das dimensões envolvidas nas interações através da teoria de conjunto. Todas as dimensões e os operadores são

pertencentes a um conjunto de sua respectiva dimensão ou de seu respectivo operador, ou seja, uma dimensão ator *(Ac)* pertence a um conjunto de todos os atores *(AcSS)*, e um operador de atividade *(atop)* pertence a um conjunto de operadores *(Ao).*

Uma sentença de interação *(Interac_Sentence$_m$)* envolve todas as dimensões. As interações são pertencentes a um conjunto de interações *(InteractionSS)* e todos os conjuntos são pertencentes ao super conjunto de interações *(InteracSS)*.

3.1.1 ATOR

Um ator é um agente que tem um papel bem definido conforme os direitos, proibições e obrigações de suas atividades. Atores são responsáveis pela execução de atividades (MARTINS & CAMOLESI JR., 2008).

As atividades podem ser individuais ou sociais, podendo assim, atingir objetos, um único ator ou grupo de atores (equipes ou comunidades) (MARTINS & CAMOLESI JR., 2008).

Os atores podem ser humanos, que são representações de pessoas reais envolvidas, e não humanos, que são "seres" virtuais envolvidos em atividades interativas com atores humanos. Todo ator (*Ac*) tem identificador (*id*), um estado corrente (*State*) e um conjunto de atributos *(AttS)*, e podem ser organizados em Grupos de Atuação (um conjunto de atores). Considerando *qh* a quantidade de atores humanos e *qs* a quantidade de atores não humanos de um ambiente, pode-se ter a dimensão Ator definida na Figura 21.

$$AchS = \{Ach_1, Ach_2, ..., Ach_{qh}\}\ ,\ AcsS = \{Acs_1, Acs_2, ..., Acs_{qs}\}$$
$$AchS \neq \varnothing \ \vee \ AcsS \neq \varnothing$$
$$AchS \cap AcsS = \varnothing$$
$$AcSS = AchS \cup AcsS$$
$$Ach_i = (Ach\_id_i\ ,\ AchState_i\ ,\ Ach\_AttS_i\ )$$
$$Acs_i = (Acs\_id_i\ ,\ AcsState_i\ ,\ Acs\_AttS_i\ )$$

Figura 21 – Dimensão Ator (CAMOLESI JR. & MARTINS, 2008).

Actors Set (AcSS) é o conjunto de todos os atores distribuído entre dois grupos disjuntos: *AchS,* que representa o conjunto de atores humanos e *AcsS,* que representa o conjunto de atores não-humanos.

Já Ach_i descreve uma tupla de ator humano, composta por identificador (Ach_id_i), estado corrente ($AchState_i$) e um conjunto de atributos (Ach_AttS_i), e Acs_i descreve uma tupla de ator não-humano na mesma composição da tupla de ator humano.

Os atores possuem atributos que são relacionados em um conjunto de atributos *(*Ach_AttS_i e Acs_AttS_i*)* . Esse conjunto descreve uma tupla composta pelo atributo e seu respectivo valor.

3.1.2 ATIVIDADE

Atividade é um elemento de execução que pode ser realizado por um ator ou grupo de atores. Atividades envolvem normalmente a manipulação ou transformação de um objeto (MARTINS & CAMOLESI JR., 2008).

Activity Super Set (AtSS) é o conjunto de todas as atividades. As Atividades *(At)* são compostas por um identificador (*id*), estado corrente (*State*), um subconjunto de atividades *(AtS)*, um subconjunto de operações *(OpS)* e um conjunto de atributos (*AttributeS*). Considerando *qa* a quantidade de atividades de um ambiente, pode-se ter a dimensão Atividade definida na Figura 22:

$$AtSS = \{At_1, At_2, \ldots, At_{qa}\}$$
$$At_i = (At\_id_i, AtState_i\ AtS_i, OpS_i, At\_AttributeS_i)$$
$$AtS_i \subseteq AtSS$$
$$OpS_i \subseteq OpS$$

Figura 22 – Dimensão Atividade (MARTINS & CAMOLESI JR., 2008).

As atividades possuem atributos que são relacionados em um conjunto de atributos *(*$At\_AttributeS_i$*)*. Esse conjunto descreve uma tupla composta pelo atributo e seu respectivo valor.

Atividades devem ser expressas em interações usando Operadores de Atividade *(atop)* que permitem a definição do direito, dever, dispensa ou proibição. Operadores de Atividade são requeridos para especificar a interação de uma atividade entre atores e objetos.

A Figura 23 mostra os operadores de atividades que pertencem ao conjunto *(Ao)*.

$$atop \in Ao = \{ right, prohibition, obligation, dispensation \}$$

Figura 23 – Dimensão Atividade - Operador (CAMOLESI JR. & MARTINS, 2005).

3.1.3 OBJETO

O conceito de Objeto é muito comum em ambientes computacionais. Objeto pode ser considerado qualquer coisa diferente de uma ação sobre objetos ou atores.

Um objeto carrega consigo a representação de suas características estruturais e de seu comportamento. Atividades e Operações podem ser realizadas sobre objetos podendo alterar suas características e estado (MARTINS & CAMOLESI JR., 2008).

Object Super Set (ObSS) é o conjunto de todos os objetos *(Ob)* usados ou manipulados. Os objetos *(Ob_i)* são compostos por um identificador *(Ob_id_i)*, um subconjunto de ObSS *($CompObS_i$)*, um estado corrente *($ObState_i$)* e um conjunto de atributos *(Ob_AttS_i)*. A Figura 24 define a dimensão Objeto.

$$ObSS = \{Ob_1, Ob_2, ..., Ob_{qo}\} \wedge ObSS \neq \varnothing$$
$$Ob_i = (Ob\_id_i, CompObS_i, ObState_i, Ob\_AttS_i)$$
$$CompObS_i \subseteq ObSS$$

Figura 24 – Dimensão Objeto (MARTINS & CAMOLESI JR., 2008).

O conjunto de atributos *(Ob_AttS_i)* descreve uma tupla composta pelo atributo e seu respectivo valor.

3.1.4 ESPAÇO

O espaço permite representar um “compartimento” ou localização de atores e objetos em ambientes, além das áreas específicas em que atividades e operações podem ocorrer. O espaço pode ser desde uma pasta de documentos em ambientes CSCL, ou mesmo elipses e polígonos em jogos eletrônicos 2D ou 3D.

Formalmente, *Space Super Set* (*SpSS*) é o conjunto de todos os espaços, e sendo *qe* a quantidade de espaços, pode-se ter a dimensão Espaço definida na Figura 25.

$$Coordinate = \{(x, y) \mid x, y \in R\}$$
$$ELoc = (Fo1, Fo2, P)$$
$$Fo1, Fo2, P \in Coordinate$$
$$PLoc = (Sp\_id_i, SpState_i, \{Po_1, Po_2, ..., Po_n\})$$
$$Po_1, ... Po_n \in Coordinate$$
$$Ellipse = \{el \mid el \in Eloc\}$$
$$Polygon = \{po \mid po \in Ploc\}$$
$$SpSS = Ellipse \cup Polygon$$
$$SpSS = \{Sp_1, Sp_2, ..., Sp_{qe}\}$$

Figura 25 – Dimensão Espaço (MARTINS & CAMOLESI JR., 2008).

As elipses (*Ellipse*) e polígonos (*Polygon*) são compostos por um identificador (*Ob_id*), um estado corrente (*SpS tate*) e coordenadas representadas por números naturais (*N*). Sendo que *Eloc* é uma tupla que descreve uma elipse e *Ploc* é uma tupla que descreve um polígono.

Os elementos da dimensão espaço devem ser expressos em interações usando um operador de espaço para a especificação de posição e tamanho de atores e objetos. A Figura 26 mostra os operadores de espaços que pertencem ao conjunto (So).

$$spop \in So = \{ <, <=, >, >=, =, <>, ==(attribution), not\ equal, inside, outside, intersect, meet, overlap, north, south, east, west\}$$

Figura 26 – Dimensão Objeto - Operador (MARTINS & CAMOLESI JR., 2008).

3.1.5 TEMPO

Diversas pesquisas têm focado a representação do tempo tendo como resultado a definição de sua taxonomia (MARTINS & CAMOLESI JR., 2008). Formalmente, *Time Super Set* (*TiSS*) é o conjunto de todos os elementos de tempo.

A formalização básica para o aspecto temporal pode ser baseada no conjunto de números naturais (*N*), para representar anos (*Ty*), meses (*Tm*), dias (*Td*), horas (*Th*), minutos (*Tmi*) e segundos (*Ts*) no Momento e Intervalo. Para Datas, conjuntos enumerados são usados para representar valores relativos (*Tmr, Tdr, Thr*, *Tmir, Tsr*) de um determinado calendário (MARTINS & CAMOLESI JR., 2008).

Considerando *qt* a quantidade de intervalos ou momentos de tempo em um ambiente, pode-se ter a dimensão Tempo definida na Figura 27.

$$Ty, Tm, Td, Th, Tmi, Ts \in N$$
$$Tmr \in \{1, 2, 3, 4, ..., 12\}$$
$$Tdr \in \{1, 2, 3, 4, 5, ..., 31\}$$
$$Thr \in \{0, 1, 2, 3, ..., 23\}$$
$$Tmir \in \{0, 1, 2, 3, ..., 59\}$$
$$Tsr \in \{0, 1, 2, 3, ..., 59\}$$
$$Time = \{Ti\_id, (Ty, Tm, Td, Th, Tmi, Ts)\}$$
$$Datetime = \{Ti\_id, (Ty, Tmr, Tdr, Thr, Tmir, Tsr)\}$$
$$Tb, Te \in Time, Interval = \{Ti\_id, (Tb, Te)\}$$
$$TiSS = Time \cup Datetime \cup Interval$$
$$TiSS = \{Ti_1, Ti_2, ..., Ti_{qt}\}$$

Figura 27 – Dimensão Tempo (MARTINS & CAMOLESI JR., 2008).

As Datas (*Datetime*), o Tempo (*Time*) e Intervalos (*Interval*) são compostos por um identificador e uma tupla que os definem.

A representação semântica do tempo permite a utilização precisa de *Operadores de tempo* para uso em expressões de interação. Baseado na modelagem de tempo, os conceitos de *duração, data* e *ocorrência* têm semânticas fundamentais para estabelecimento de referências temporais. Estas semânticas são usadas para definir a lógica temporal de operadores para tempo de duração, data de ocorrência ou intervalo de ocorrência de atividades e operações definidas em interações entre atores e objetos (MARTINS & CAMOLESI JR., 2008).

A Figura 28 mostra os operadores de tempo (*tiop*), eles são necessários para especificar as interações no tempo, intervalos ou datas.

$$tiop \in To = \{ <, <=, >, >=, =, <>, == \text{ (attribution), precedes, succeeds, directly precedes, directly succeeds, overlaps, is overlapped by, occurs during, contains, starts, is started with, finishes, is finished with, coincides with } \}$$

Figura 28 – Dimensão Tempo - Operador (MARTINS & CAMOLESI JR., 2008).

3.2 ASSOCIAÇÕES

As associações podem ser estáticas ou dinâmicas. As estáticas são definidas no começo das interações e permanecem inalteradas até o final de um processo. Já as dinâmicas sofrem alterações caso necessárias.

Em situações específicas, pode-se estabelecer ou verificar associações entre atores, atores e objetos, atores e tempo, atores e espaços, e objetos e espaços.

Entre as associações mais comuns encontram-se os tipos (MARTINS & CAMOLESI JR., 2008):

- Propriedade (possuir) - é uma associação em que um ator ou grupo de atores tem a posse de espaços, de objetos em determinados espaços, de objetos durante certo tempo, ou espaços durante certo tempo;
- Utilização (usar) - é uma associação em que um ator ou grupo de atores tem direitos de utilização de objetos em determinados espaços, objetos durante certo tempo, ou espaços durante certo tempo;
- Agrupamento é uma abstração de semântica mais flexível que a composição/decomposição, pois permite que atores, objetos, espaços, períodos ou momentos de tempo, atividades ou operações possam ser agrupados em conjuntos homogêneos ou heterogêneos independentemente de qualquer orientação física ou lógica.

3.3 ABSTRAÇÕES

A prática de abstração é bastante conhecida em processos de modelagem de sistemas, onde através de um modelo busca-se representar da maneira mais adequada os elementos de um ambiente e seus relacionamentos.

As abstrações de maior ocorrência em ambientes colaborativos são a classificação, generalização e composição (MARTINS & CAMOLESI JR., 2008).

- Classificação - está presente em qualquer ambiente colaborativo, pois permite a especificação de tipos (ou seja, atores, objetos, atividades, tempos e espaços) e instâncias na formação de expressões de regras que genericamente envolvem todos os elementos de um tipo ou um elemento específico. A classificação é uma construção semântica que permite a elaboração de uma hierarquia de classificação em que tipos podem ser instanciados ao mesmo tempo em que são instâncias de supertipos.
- Generalização – podem-se representar aspectos estruturais, funcionais e comportamentais, genéricos e específicos de atores, objetos, tempos, atividades e espaços. Em expressões de regras, esta abstração permite estabelecer normas de interação gerais e particularizadas. A generalização é uma construção semântica que permite a elaboração de uma hierarquia de generalização, em que tipos podem ter subtipos que ainda podem ter outros

subtipos e assim por diante, em um processo de especificação até um grau de refinamento necessário para a legibilidade e consistência das regras de colaboração.

- Composição é usualmente empregada para indicar que dois ou mais objetos, espaços, períodos ou momentos de tempo estão associados física ou logicamente como partes de um elemento complexo. Esta abstração é representada através de referências entre os elementos envolvidos. A composição é uma construção semântica que permite a elaboração de uma hierarquia de composição em que os elementos da ontologia podem ser decompostos em um processo de refinamento necessário para a legibilidade e consistência das regras de colaboração.

3.4 POLÍTICAS DE COLABORAÇÃO

Políticas de Colaboração definem normas para as possíveis interações em um ambiente, podendo ser estabelecidas por agrupamento de regras com metas ou objetivos bem definidos.

A definição destas políticas em sistemas pode ser de duas formas (MARTINS & CAMOLESI JR., 2008):

- Intra-regras: em que a política é definida internamente às regras de um ambiente, ou seja, na definição de uma regra podem existir referências a outras regras que possuam algum grau de interdependência e assim, as associações entre regras que caracterizam uma política estão estabelecidas internamente ao corpo das regras;
- Extra-regras: em que a política é definida externamente às regras, e desta forma todas as inter-relações entre regras que estabelecem uma composição para normatização de um ambiente são definidas em um elemento independente, no caso, podendo ser a própria Política.

O modelo *M-Forum* define políticas na forma de intra-regras, como sendo o conjunto de regras ativas. A figura 29 apresenta um modelo conceitual dos principais elementos segundo o M-Forum.

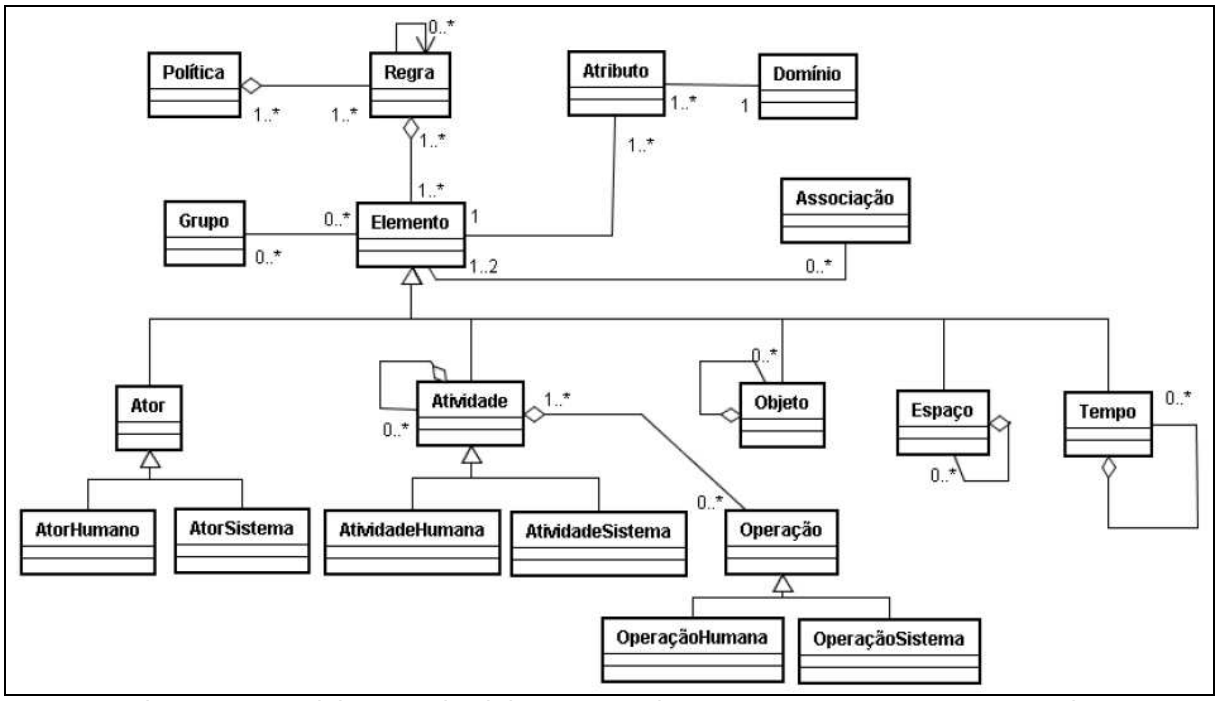

Figura 29 – Modelo Conceitual do M-Forum (MARTINS & CAMOLESI JR., 2008).

3.5 LINGUAGEM L-FORUM

A Linguagem *L-Forum* é uma linguagem formal (Anexo) para definição de regras que auxilia a governabilidade de um ambiente colaborativo, podendo trazer diversos benefícios, entre os quais pode-se citar (BRESSAN, 2008):

- Orientar na definição de regras do ambiente;
- Aumentar o grau de precisão das regras, diminuindo a ambigüidade na interpretação pelos atores do ambiente;
- Facilitar a depuração de inconsistências;
- Permitir a flexibilidade do ambiente com a retirada, modificação ou inclusão de regras.

A Linguagem *L-Forum* também possui três conjuntos de cláusulas com propósitos bem específicos (BRESSAN, 2008):

- *Contexto*: composto por *parâmetros* para a execução ou ativação de uma regra e pelas condições de aplicabilidade que estabelecem os cenários (valores de atributos, aspectos temporais ou espaciais) em que uma regra pode ser aplicada;

- *Definição (ou corpo)*: conjunto de expressões que estabelece as ações ou condições para as ações que interagem entre os elementos, podendo envolver opcionalmente aspectos de transitoriedade no tempo e no espaço;

- *Regime*: é um item opcional composto pelo conjunto de regras inter-relacionadas que tenham orientação para serem executadas ou aplicadas. Também podem ser definidos os cenários (valores de atributos, aspectos temporais ou espaciais) em que uma regra deve ser ativada ou desativada para uso.

Para exemplificar a aplicação da Linguagem *L-Forum* na especificação de regras, a abstração será a travessia de veículos e pedestres em um cruzamento de vias. A sinalização pode ser regida por um semáforo, ou na falta dele, por placas de regulamentação.

Tabela 2: Alguns elementos e associações envolvidos em um cruzamento de vias.

Atores	Motorista, Pedestre
Objetos	Veículo, Semáforo, Placa de Sinalização, Faixa de Pedestre, Calçadas
Grupo	Pedestres, Veículos
Atividades	Identificar sinalização da via, Identificar cores do semáforo, obedecer, parar, prosseguir, aguardar, dar a preferência.
Tempo	Tempo do semáforo
Espaço	Vias de trânsito de veículos, Faixa de Pedestre, Calçadas
Abstrações	Veículos.transita.Vias Pedestre.transita.calçadas Pedestre.transita.pela_faixa_de_pedreste Vias.possui.placa_de_sinalização Vias.possui.semáforo Semáforo.possui.cores Veículo.possui.motorista Motorista.identifica.placa_de_sinalização Motorista.identifica.cores_do_semáforo Pedestre.identifica.placa_de_sinalização Pedestre.identifica.cores_do_semáforo

Na Tabela 2 são identificados alguns elementos e associações encontrados em um cruzamento de vias. Cada elemento é distribuído conforme sua dimensão, e são listadas algumas associações abstraídas neste exemplo. Na tabela 3 são especificadas algumas regras previstas para a situação de travessia, cada uma com um nome identificador. Depois seu contexto é definido utilizando-se as cláusulas *Parameters* e *Applicability*.

Em *Parameters* pode-se definir qualquer elemento estrutural de um ambiente. Os elementos estruturais reconhecidos pela linguagem são: *actor, group, object, activity, operation, space, time e association* (Anexo).

Em Applicability, define-se em que condição a regra pode ser aplicada; portanto nesta cláusula, uma expressão condicional sempre deverá ser avaliada. Na cláusula *Body* define-se o corpo da regra, especificando-se as condições de funcionamento interno da regra, as ações embutidas na regra e possíveis invocações de outras regras (encadeamento de regras).

Tabela 3: Exemplo de Regras em L-Forum

```
Rule          Identificar Sinalização das Vias          [Ative]
{
Parameters::  (m:motorista, p:pedestres, pa:placa de sinalização, v:vias, s:semaforo)
Applicability:: (m ON v) OR (p ON v)
Body::        Execute a Action (m obligation identifica pa)
              Execute a Action (p obligation identifica pa)
              IF (s NOT IN v)
                  THEN Rule (Obedecer Placa de Sinalização (m, p, pa, v, vt:via_transversal, ve:veiculo) right)
                  ELSE Rule (Obedecer Sinalização do Semaforo (m, p, s, v, c:cores_do_semaforo) right)
}
Rule          Obedecer Placa de Sinalização          [Ative]
{
Parameters::  (m:motorista, p:pedestres, pa:placa de sinalização, v:vias, vt:via_transversal, ve:veiculo)
Applicability:: (m ON v) OR (p ON v)
Body::        Execute a Action (m obligation verifica pa)
              Execute a Action (p obligation verifica pa)
              IF (pa = 'stop')
                  THEN Action (m obligation parar ve)
                          Action (ve right preferencia vt)
                          Action (p right preferencia vt)
                   ELSE Action (m right preferencia v)
                          Action (p obligation dar_preferencia vt)
}
Rule          Obedecer sinalização do semaforo          [Ative]
{
Parameters::  (m:motorista, p:pedestres, v:vias, s:semaforo, c:cores_do_semaforo)
Applicability:: (m ON v) OR (p ON v)
Body::        Execute a Action (m obligation identifica s)
              Execute a Action (p obligation identifica s)
              IF (c = 'green')
                 THEN Action (m rigth transitar v)
                         Action (p obligation parar v)
              IF (c = 'yellow')
                  THEN Action (m obligation atenção s)
                          Action (p obligation parar v)
              IF (c = 'red')
                  THEN Action (m obligation parar v)
                          Action (p rigth transitar v)
}
```

A primeira regra exemplificada na Tabela 3 é identificada pelo nome identificador *Identificar Sinalização das Vias.* Os paramentos para essa regra são: motorista, pedestres, placa de sinalização, vias e semáforo. Esta regra se aplica para todos os motoristas e pedestres em cruzamento de vias de trânsito. No corpo de definição eles têm por obrigação identificar a sinalização da via, neste caso, a

existência de um semáforo ou não. A não existência de um semáforo faz a invocação da regra *Obedecer Placa de Sinalização*, caso contrário, será feita a invocação da regra *Obedecer Sinalização do Semáforo*.

A segunda regra exemplificada na Tabela 3 é invocada pelo nome identificador *Obedecer Placa de Sinalização,* e tem como parâmetros: motorista, pedestres, placa de sinalização, vias, via transversal e veículo. Esta regra se aplica para todos os motoristas e pedestres em cruzamento de vias de trânsito. No corpo de definição eles têm por obrigação identificar a sinalização da via, neste caso, uma placa de parada obrigatória. A existência da placa de parada obrigatória obriga o motorista a ter a ação de parar o veículo, dando o direito de preferência para o veículo que transita pela via transversal e para os pedestres. A não existência da placa de parada obrigatória o motorista tem o direito a preferência no cruzamento, ficando ao pedestre a obrigação de parar.

A terceira regra exemplificada na Tabela 3 é invocada pelo nome identificador *Obedecer Sinalização do Semáforo*. Os parâmetros para essa regra são: motorista, pedestres, vias, semáforo, cores do semáforo. Esta regra se aplica para todos os motoristas e pedestres em cruzamento de vias de trânsito. No corpo de definição eles têm por obrigação identificar a sinalização da via, neste caso, um semáforo. As cores do semáforo passa a reger as ações que serão invocadas. Caso a cor do semáforo seja verde (*green*) o motorista tem o direito de transitar pela via e o pedestre a obrigação de esperar (parar). Caso a cor do semáforo seja amarelo (*yellow*), o motorista tem a obrigação de ter atenção ao semáforo e o pedestre a ter a obrigação de esperar (parar). Caso a cor do semaforo seja vermelho (*red*) o motorista tem a obrigação de parar e o pedestre tem o direito de transitar pela via.

3.6 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os conceitos que define o modelo de especificação de regras *M-Forum*, as definições de suas Dimensões através da teoria de conjuntos, as Associações, as Abstrações e a Política de Colaboração.

Para mostrar a aplicabilidade da linguagem *L-Forum*, foram especificadas algumas regras, mostrando sua estrutura e sintaxe. Contudo, foi possível perceber a necessidade de se criar uma maneira de especificar as

dimensões e associações que são encontradas nas regras especificadas pela *L-Forum*. Essa necessidade motivou a criação de uma linguagem que especificasse ontologicamente as dimensões e associações envolvidas nas interações, especificadas pelo modelo *M-Forum*.

CAPÍTULO 4

A LINGUAGEM ONTOL-FORUM

Para incorporar recurso de especificação de ontologias na linguagem *L-Forum*, este trabalho realizou o estudo e a especificação de uma extensão da linguagem que recebeu a denominação *OntoL-Forum*. A *OntoL-Forum* foi criada com o objetivo de permitir a definição ontológica dos elementos envolvidos nas interações especificadas pela linguagem *L-Forum*, criando uma estrutura simples que possa ser facilmente entendida e compartilhada.

A linguagem possui uma semântica semelhante à *Ontolingua* e uma estrutura sintética (vide apêndice, à página 68) igual a *L-Forum*, mas é independente para a definição das ontologias. Sua sintaxe lembra a linguagem KEE (GRUBER, 1993), que traduz *Ontolingua* para um sistema de conhecimento, mas não é tão expressiva.

4.1 A SEMÂNTICA

Como na *Ontolingua* a linguagem *OntoL-Forum* é constituída por: um nome, uma lista de argumentos, string de declarações e sentenças. Em *OntoL-Forum* nem todos os termos que compõem o corpo de definição são obrigatórios na definição de uma classe, exceto: o nome, a instância, a dimensão e a declaração sobre a ontologia que está sendo definida. Também podem ser definidas classes abstratas, utilizando a palavra-chave '*abstract*' no local do termo instância.

```
Ontology-Class nome_ontologia [instância]
{
   Dimension: nome_dimensão
   Comment: 'declarações_sobre_a_ontologia'
   ImportOntology: 'menção_a_outra_ontologia'
   SuperClass: classe_superior
   Property: atributos_e_restrições_de_atributos
   ConstraintOntology: restrições_sobre_ a_ classe_ definida
}
```

Figura 30 – Corpo de Definição para Classe em *OntoL-Forum*.

A Figura 30 mostra o corpo de definição da linguagem *OntoL-Forum.* O campo *Ontology-Class* define o nome da ontologia que vai ser especificada e sua instância. O campo *Dimension* faz menção a que dimensão a ontologia faz parte. No campo *Comment* é onde são feitos os comentários conceituais sobre a ontologia.

O campo *ImportOntology* faz menção a uma ontologia disponibilizada na *Web* ou criada pela própria linguagem. O campo *SuperClass* faz menção à classe superior em que a ontologia a ser definida está submetida. No campo *Property* são definidos os atributos e suas restrições, como por exemplo, a cardinalidade. Já o campo *ConstraintOntology* faz menção a alguma restrição na ontologia que está sendo definida.

A Figura 31 mostra a definição da classe Author em *Ontolingua*, exemplificada no capítulo 2. A Figura 32 mostra sua definição em *OntoL-Forum*.

```
(define-class AUTHOR (?author)
   "An author is a person who writes things. An author must have created at least one document.
   In this ontology, an author is known by his or her real name."
      :def (and (person ?author)
                (= (value-cardinality ?author AUTHOR.NAME) 1)
                (value-type ?author AUTHOR.NAME biblio-name)
                (>= (value-cardinality ?author AUTHOR.DOCUMENTS) 1)
                (<=> (author.name ?author ?name)
                     (person.name ?author ?name))))
```

Figura 31 – Definição da classe Author em *Ontolingua* (GRUBER, 1993).

Em *OntoL-Forum*:

```
Ontology-Class Author [author]
{
   Dimension: actor
   Comment: 'An author is a person who writes things. An author must have created at least
            one document. In this ontology, an author is known by his or her real name'.
   ImportOntology: 'http://www.libraryontology.com/autor'
   SuperClass: Person
   Property: Author.Name
      ValueClass: (fieldOf Biblio.name)
      MinCard: 1
      MaxCard: 1
    Property: Author.Documents
      MinCard: 1
    ConstraintOntology: (author.name = person.name)
}
```

Figura 32 – Definição da classe Author em *OntoL-Forum*.

O campo *Property* é o único que se repete em *OntoL-Forum* conforme o número de atributos. A ontologia Author pertence à dimensão actor.

4.2 A ONTOLOGIA DAS DIMENSÕES

As dimensões são elementos envolvidos nas interações entre seres (atores) e objetos dentro de um ambiente (espaço) formalizado por um período de tempo em que as atividades devem ser desenvolvidas.

No modelo *M-Forum* as cinco dimensões relacionadas em uma interação são representadas por atores, atividades, objetos, tempo e espaço, com o envolvimento de operadores para as respectivas dimensões e as associações entre eles. As ontologias definidas em *OntoL-Forum* são específicas para o modelo de regras *M-Forum* e as regras definidas por ele.

4.2.1 A DIMENSÃO ATOR

Um ator é um agente que tem um papel bem definido conforme os direitos, proibições e obrigações de suas atividades e são responsáveis pela execução de atividades (CAMOLESI JR. & MARTINS, 2005).

Conforme a formalização matemática da dimensão Ator apresentada na Figura 21, página 30, a Figura 33 mostra a definição ontológica da dimensão Ator em *OntoL-Forum*:

```
Ontology-Class Actor [abstract]
{
   Dimension: Actor
   Comment: 'Um ator é um agente que tem um papel bem definido conforme os direitos,
            proibições e obrigações de suas atividades e são responsáveis pela execução de
            atividades. Os atores podem ser humanos e não humanos distribuídos entre dois
            grupos disjuntos. Atores não humanos são seres virtuais envolvidos em
            atividades interativas com atores humanos. Os atores podem ser organizados
            em grupos de atuações. '
   ImportOntology: ''
   SuperClass:
   Property: Actor.Identification
        Value: Integer
        MinCard: 1
        MaxCard: 1
   Property: Actor.State
        ValueClass: (fieldOf StateMachineSet)
        MinCard: 1
   Property: Actor.Attributes
        Value: Attributes = {(name, value)}
        MinCard: 1
   ConstraintOntology: ()
}
```

Figura 33 – Definição da Dimensão Ator em *OntoL-Forum*.

A ontologia apresentada na Figura 33 define a dimensão Ator, sendo uma classe abstrata por não possuir instância e é definida contendo um identificador (*Actor.Identification*), um estado (*Actor.State*) e atributos (*Actor.Attributes*). No campo *'Comment'* é feita uma definição conceitual da dimensão, e os outros campos definem as condições necessárias e suficientes para sua existência.

Neste caso, no atributo *Ator. Identification* se encontra uma restrição de valor e de cardinalidade, significando que a dimensão deve possuir obrigatoriamente um identificador, e somente um, sendo seu valor do tipo inteiro.

No atributo *Ator.State* a palavra-chave *ValueClass* mostra que os valores para esse atributo são valores definidos na classe *StateMachineSet*, nos mostrando uma restrição de valor. Já sua cardinalidade mostra que todo ator possui no mínimo um estado corrente.

Já os valores para o atributo *Ator.Attributes* é um conjunto de atributos indicado pela palavra-chave '={ }', sendo que cada atributo é uma tupla com o atributo e seu respectivo valor entre parênteses. O conjunto de atributos é finito e pode ter mais de uma tupla. A cardinalidade desse atributo significa que a dimensão possui no mínimo um atributo.

4.2.2 A DIMENSÃO ATIVIDADE

Atividade é um elemento de execução que pode ser realizado por um ator ou grupo de atores. Atividades envolvem normalmente a manipulação ou transformação de um objeto (CAMOLESI JR. & MARTINS, 2005).

A ontologia apresentada na Figura 34 define a dimensão Atividade, assim como a dimensão Ator, Atividade é uma classe abstrata por não possuir instância. É definida como algo que possui identificador (*Activity.Identification*), estado (*Activity.State*), atividades (*Activity.Activities*), operações (*Activity.Operations*), atributos (*Activity.Attributes*) e operador (*Activity.Operator*). No campo *'Comment'* é feita a definição conceitual da dimensão, e os outros campos definem as condições necessárias e suficientes para sua existência.

Neste caso, os atributos *Activity.Identification, Activity.State* e *Activity.Attributes* têm significados semelhantes aos da ontologia Ator. Os atributos *Activity.Activities* e *Activity.Operations* têm como valor um conjunto de atividades e operações respectivamente, indicado pela palavra-chave '={ }'. A cardinalidade

definida para eles significa que pelo menos uma atividade, e uma operação, existem nesta dimensão.

Já os valores para o atributo *Activity.Operator* são definidos na classe *ActivityOperationSet*, e sua cardinalidade significa que a dimensão deve conter no mínimo um operador.

Conforme a formalização matemática da dimensão Atividade apresentada nas Figuras 22 e 23, página 31, a Figura 34 mostra a definição ontológica da dimensão Atividade em *OntoL-Forum*:

```
Ontology-Class Activity [abstract]
{
   Dimension: Activity
   Comment: 'Atividade é um elemento de execução que pode ser realizado por um ator ou
            grupo de atores. Atividades envolvem normalmente a manipulação ou
            transformação de um objeto.
            As atividades devem ser expressas em interações usando operadores de
            atividade que permitem a definição do direito, dever, dispensa ou proibição.
            Operadores de atividade são requeridos para especificar a interação de uma
            atividade entre atores e objetos. '
   ImportOntology: ''
   SuperClass:
   Property: Activity.Identification
        Value: Integer
        MinCard: 1
        MaxCard: 1
   Property: Activity.State
        ValueClass: (fieldOf StateMachineSet)
        MinCard: 1
   Property: Activity.Activities
        Value: Activities = {Activity}
        MinCard: 1
   Property: Activity.Operations
        Value: Operations = {Operation}
        MaxCard: 1
   Property: Activity.Attributes
        Value: Attributes = {(name, value)}
        MinCard: 1
   Property: Activity.Operator
        ValueClass: (fieldOf ActivityOperationSet)
        MinCard: 1
   ConstraintOntology: ()
}
```

Figura 34 – Definição da Dimensão Atividade em *OntoL-Forum*.

4.2.3 A DIMENSÃO OBJETO

Objetos representam elementos que constituem conceitos ou entidades do mundo real. Um objeto carrega consigo a representação de suas características estruturais e de seu comportamento (MARTINS & CAMOLESI JR., 2008).

Conforme a formalização matemática da dimensão Objeto apresentada na Figura 24, página 32, a Figura 35 mostra a definição ontológica da dimensão Objeto em *OntoL-Forum*:

```
Ontology-Class Object [abstract]
{
   Dimension: Object
   Comment: 'Um objeto carrega consigo a representação de suas características estruturais e
             de seu comportamento. Atividades e Operações podem ser realizadas sobre
             objetos podendo alterar suas características e estado. Objetos podem ser
             compostos por outros objetos. '
   ImportOntology: ''
   SuperClass:
   Property: Object.Identification
       Value: Integer
       MinCard: 1
       MaxCard: 1
   Property: Object.Components
       Value: Components = {(component)}
       MinCard: 1
   Property: Object.State
       ValueClass: (fieldOf StateMachineSet)
       MinCard: 1
   Property: Object.Attributes
       Value: Attributes = {(name, value)}
       MinCard: 1
   ConstraintOntology: ()
}
```

Figura 35 – Definição da Dimensão Objeto em *OntoL-Forum*.

A ontologia apresentada na Figura 35 define a dimensão Objeto. Uma classe abstrata como as outras dimensões já definidas anteriormente, por não possuir instância. É definida como algo que possui identificador (*Object.Identification*), componentes (*Object.Components*), estado (*Object.State*) e atributos (*Object.Attributes*). No campo *'Comment'* é feita a definição conceitual da dimensão, e os outros campos definem as condições necessárias e suficientes para sua existência.

Neste caso, os atributos *Object.Identification, Objeto.State* e *Objeto.Attributes* têm significados semelhante às ontologias anteriormente definidas. O atributo *Object.Components* tem como valor um conjunto de componentes indicado pela palavra-chave '={ }', e sua cardinalidade significa que deve conter no mínimo um componente nesta dimensão. Um objeto pode conter outros objetos na sua composição, ou mesmo ser formado por um somente, neste caso, ele mesmo.

4.2.4 A DIMENSÃO ESPAÇO

O conceito de espaço permite representar um “compartimento” ou localização de atores e objetos em ambientes, além das áreas específicas em que atividades e operações podem ocorrer (MARTINS & CAMOLESI JR., 2008).

Conforme a formalização matemática da dimensão Espaço apresentada nas Figura 25 e 26, página 33, a Figura 36 mostra a definição ontológica da dimensão Espaço em *OntoL-Forum*:

```
Ontology-Class Space [abstract]
{
   Dimension: Space
   Comment: 'O espaço permite representar um "compartimento" ou localização de atores e
            objetos em ambientes, além das áreas específicas em que atividades e
            operações podem ocorrer. O espaço pode ser desde uma pasta de documentos
            em ambientes CSCL, ou mesmo, elipses e polígonos em jogos eletrônicos 2-D.
            Elementos da dimensão espaço devem ser expressos em interações usando o
            Operador de Espaço para a especificação de posição e tamanho de atores e
            objetos. '
   ImportOntology: ''
   SuperClass:
   Property: Space.Identification
       Value: Integer
       MinCard: 1
       MaxCard: 1
   Property: Space.State
       ValueClass: (fieldOf StateMachineSet)
       MinCard: 1
   Property: Space.Coordinates
       Value: Coordinates = {(x,y) | localization}
       MinCard: 1
   Property: Space.Operator
       ValueClass: (fieldOf SpaceOperationSet)
       MinCard: 1
   ConstraintOntology: ()
}
```

Figura 36 – Definição da Dimensão Espaço em *OntoL-Forum*.

A ontologia apresentada na Figura 36 define a dimensão Espaço. Uma classe abstrata como as outras dimensões já definidas anteriormente, por não possuir instância. É definida como algo que possui identificador (*Space.Identification*), estado (*Space.State*), coordenadas (*Space.Coordinates*) e operador (*Space.Operator*). No campo *‘Comment’* é feita a definição conceitual da dimensão, e os outros campos definem as condições necessárias e suficientes para sua existência.

Neste caso, os atributos *Space.Identification, Space.State* e *Space.Operator* tem significados semelhantes às ontologias anteriormente definidas.

O atributo *Space.Coordinates* tem como valor um conjunto de tuplas de coordenadas ou de endereço de pasta de documentos em ambientes CSCL[20], indicado pela palavra-chave '={ }', e sua cardinalidade significa que deve conter no mínimo um subconjunto nesta dimensão.

4.2.5 A DIMENSÃO TEMPO

A formalização básica para o aspecto temporal foi baseada no conjunto de números naturais (N), para representar anos, meses, dias, horas, minutos e segundos no Momento e Intervalo. Para as Datas, conjuntos enumerados são usados para representar valores relativos de um determinado calendário (MARTINS & CAMOLESI JR., 2008).

A dimensão Tempo é dividida em três tipos excludentes de Tempo: Momento, Duração e Intervalo. Por esse motivo sua ontologia será representada por uma hierarquia de classes abstratas, em que cada tipo excludente será definido ontologicamente como subclasse da dimensão Tempo.

O tipo Momento trata do tempo absoluto, o tempo no instante em que uma atividade se realiza. O tipo Duração trata do tempo relativo, ou seja, o prazo em que se estabelece a aplicabilidade da regra. Já o Intervalo, é o tempo entre dois momentos, ou seja, o intervalo em que a regra se aplica.

Conforme a formalização matemática da dimensão Tempo apresentada nas Figura 27 e 28, página 34, as Figuras 37, 38, 39 e 40 mostram a definição ontológica da dimensão Tempo e seus tipos excludentes em *OntoL-Forum*:

```
Ontology-Class Time [abstract]
{
  Dimension: Time
  Comment: 'O Tempo tem uma formalização básica baseada no conjunto de números
           naturais (N) para representar anos, meses, dias, horas, minutos e segundos no
           Momento e Intervalo. Para Datas, conjuntos enumerados são usados para
           representar valores relativos de um determinado calendário. A representação
           semântica do tempo permite a utilização precisa de operadores de tempo para
           uso em expressões de interação em ambientes. '
  ImportOntology: ''
  SuperClass:
  Property: Time.Operator
      ValueClass: (fieldOf TimeOperationSet)
      MinCard: 1
  ConstraintOntology:
}
```

Figura 37 – Definição da Dimensão Tempo em *OntoL-Forum*.

[20] *CSCL (Computer Supported Collaborative Learning)* – Aprendizagem Colaborativa Apoiada por Computador pode ser definida como um modelo aprendizado em que o conhecimento é construído por dois ou mais indivíduos através da discussão, da reflexão e tomada de decisões, tendo como mediador desse processo os recursos computacionais (Internet, dentre outros).

A ontologia apresentada na Figura 37 define a dimensão Tempo. Uma classe abstrata como as outras dimensões já definidas anteriormente, por não possuir instância. É definida conceitualmente no campo *'Comment'* e possui somente um atributo, o *Time.Operator*. Os valores para esse atributo são definidos na classe *TimeOperationSet*, e sua cardinalidade significa que a dimensão deve conter no mínimo um operador.

A ontologia apresentada na Figura 38 define a dimensão Momento. Esta dimensão é subclasse da dimensão Tempo, sendo uma classe abstrata por não possuir instância. No campo *'Comment'* é feita a definição conceitual da dimensão, e os outros campos definem as condições necessárias e suficientes para sua existência. O atributo *Moment.Instant* tem como valor uma tupla que possui um identificador com uma outra tupla para determinados valores para expressar o tempo absoluto, ou seja, o tempo no instante em que a atividade se realiza. A cardinalidade para esse atributo significa que esta dimensão contém somente um valor para ele.

Já os valores para o atributo *Moment.Operator* são definidos na classe *TimeOperationSet*, e sua cardinalidade significa que a dimensão deve conter no mínimo um operador. Nesta dimensão o campo *ConstraintOntology* é utilizado e faz menção às restrições de valores, e a sentença (*Moment.Operator* = *Time.Operator*) significa que ambos são os mesmos.

```
Ontology-Class Moment [abstract]
{
   Dimension: Time
   Comment: 'A ontologia Momento é uma subclasse da Ontologia Time. É um tipo excludente
            de tempo que trata do tempo absoluto, ou seja, o tempo no instante em que a
            atividade se realiza. '
   ImportOntology: ''
   SuperClass: Time
   Property: Moment.Instant
      Value: Instant = (identification, (years, months, days, hours, minutes, seconds))
      MinCard: 1
      MaxCard: 1
   Property: Moment.Operator
      ValueClass: (fieldOf TimeOperationSet)
      MinCard: 1
   ConstraintOntology: (  months <= 12
                          days <= 31
                          hours <= 24
                          minutes <= 59
                          seconds <= 59
                          Moment.Operator  = Time.Operator   )
}
```

Figura 38 – Definição da Dimensão Momento em *OntoL-Forum*.

A ontologia apresentada na Figura 39 define a dimensão Duração. Esta dimensão também é subclasse da dimensão Tempo, sendo uma classe abstrata por não possuir instância. No campo *'Comment'* é feita a definição conceitual da dimensão, e os outros campos definem as condições necessárias e suficientes para sua existência.

O atributo *Duration.Period* tem como valor uma tupla que possui um identificador com uma outra tupla para determinados valores para expressar o tempo relativo, ou seja, o prazo em que se estabelece a aplicabilidade da regra. A cardinalidade para esse atributo significa que esta dimensão contém somente um valor para ele.

Já os valores para o atributo *Duration.Operator* são definidos na classe *TimeOperationSet*, e sua cardinalidade significa que a dimensão deve conter no mínimo um operador. Nesta dimensão o campo *ConstraintOntology* também é utilizado e faz menção às restrições de valores, e a sentença (*Duração.Operador* = *Time.Operador*) significa que ambos são os mesmos.

```
Ontology-Class Duration [abstract]
{
   Dimension: Time
   Comment: 'A ontologia Duração é uma subclasse da Ontologia Tempo. É um tipo excludente
            de tempo que trata do tempo relativo, ou seja, o prazo em que se estabelece a
            aplicabilidade da regra. '
   ImportOntology: ''
   SuperClass: Time
   Property: Duration.Period
      Value: Period = (identification, (years, months, days, hours, minutes, seconds))
      MinCard: 1
      MaxCard: 1
   Property: Duration.Operator
      ValueClass: (fieldOf TimeOperationSet)
      MinCard: 1
   ConstraintOntology: (  months <= 12
                          days <= 31
                          hours <= 24
                          minutes <= 59
                          seconds <= 59
                          Duration.Operator = Time.Operator  )
}
```

Figura 39 – Definição da Dimensão Duração em *OntoL-Forum*.

A ontologia apresentada na Figura 40 define a dimensão Intervalo. Esta dimensão também é subclasse da dimensão Tempo, sendo uma classe abstrata por não possuir instância. No campo *'Comment'* é feita a definição

conceitual da dimensão, e os outros campos definem as condições necessárias e suficientes para sua existência.

O atributo *Interval.Intervals* tem como valor uma tupla que possui um identificador com uma outra tupla para determinados valores para expressar o tempo entre dois momentos. A cardinalidade para esse atributo significa que esta dimensão contém somente um valor para ele.

```
Ontology-Class Interval [abstract]
{
   Dimension: Time
   Comment: 'A ontologia Intervalo é uma subclasse da Ontologia Tempo. É um tipo excludente
            de tempo onde envolve dois Momentos, ou seja, o intervalo em que a regra se
            aplica. '
   ImportOntology: ''
   SuperClass: Time
   Property: Interval.Intervals
      Value: Intervals = (identification, (time01, time02))
      MinCard: 1
      MaxCard: 1
   Property: Interval.Operator
      ValueClass: (fieldOf TimeOperationSet)
      MinCard: 1
   ConstraintOntology: (  Intervals.time01 = Moment
                          Intervals.time02 = Moment
                          Intervals.time01 ≠ Intervals.time02
                          Interval.Operator  = Time.Operator        )
}
```

Figura 40 – Definição da Dimensão Intervalo em *OntoL-Forum*.

Já os valores para o atributo *Interval.Operator* são definidos na classe *TimeOperationSet*, e sua cardinalidade significa que a dimensão deve conter no mínimo um operador. Nesta dimensão o campo *ConstraintOntology* também é utilizado e faz menção as restrições de valores, e a sentença (*Intervalo.Operador* = *Time.Operador*) significa que ambos são os mesmos.

4.3 ASSOCIAÇÕES

A associação é um tipo de relacionamento que no modelo para especificação de regras *M-Forum* pode ocorrer entre atores, atores e objetos, atores e tempo, atores e espaços, e objetos e espaços. Estas associações são estáticas se permanecerem inalteradas até o final de um processo, ou são dinâmicas caso sejam alteradas durante uma interação.

Assim como em *Ontolingua* as classes, relações e funções são definidas com a mesma sintaxe apoiada pelos mesmos campos de sentenças, o mesmo acontece em *OntoL-Forum*. A diferença entre uma classe e uma associação

em *OntoL-Forum* se faz no campo instância, substituído pelo campo argumentos, sendo permitido mais que um argumento neste campo para a definição de uma associação. Também o campo *Dimension* é substituído pelo campo *Abstraction* onde é tipificada a abstração conforme a especificação do modelo M-Forum no capítulo 3, página 29.

A Figura 41 mostra o corpo de definição para associação em *OntoL-Forum*:

```
Ontology-Association nome_associação [argumentos]
{
   Abstraction: tipo_abstração
   Comment: 'declarações_sobre_a_ontologia'
   ImportOntology: 'menção_a_outra_ontologia'
   SuperClass: classe_superior
   Property: atributos_e_restrições_de_atributos
   ConstraintOntology: restrições_sobre_a_classe_definida
}
```

Figura 41 – Corpo de Definição para Associação em *OntoL-Forum* .

A Figura 42 mostra a definição de uma ontologia para a associação Possui, exemplificada no Capítulo 3, página 38, tabela 2. A ontologia que define esta associação tem como paramentos os argumentos *veículo* e *motorista*. O campo '*Abstration*' é tipificado como uma associação de propriedade (Porperty). No campo *'Comment'* é feita a definição conceitual da associação, e os outros campos definem as condições necessárias e suficientes para sua existência.

```
Ontology-Association Possui [veículo motorista]
{
   Abstration: Property
   Comment: 'Um veículo é conduzido por um motorista em vias de trânsitos.'
   ImportOntology: ''
   Superclasse:
   Property: Possui.Veículo
        ValueClass: (intanceOf Veículo)
        MinCard: 1
        MaxCard: 1
   Property: Possui.Motorista
        ValueClass: (intanceOf Motorista)
        MinCard: 1
        MaxCard: 1
   ConstraintOntology:
}
```

Figura 42 – Definição de uma associação em *OntoL-Forum* .

Neste caso, é possível notar os atributos *Possui.Veículo* e *Possui.Motorista*. O atributo *Possui.Veículo* é uma instância da classe Veículo, e sua cardinalidade mostra que esta associação envolve apenas um veículo. Já o atributo *Possui.Motorista* é uma instância da classe Motorista, e sua cardinalidade também mostra que esta associação envolve um motorista. Assim como na definição de uma classe não há necessidade de usar todos os campos do corpo de definição para sua especificação, o mesmo acontece na especificação de uma associação.

4.4 Estudo de Caso

Neste tópico são definidas algumas ontologias referentes às regras apresentadas no Capítulo 3, na exemplificação da linguagem *L-Forum*. A abstração apresentada foi a travessia de veículos e pedestres em um cruzamento de vias. Nesse contexto as regras servem para indicar direito e a obrigações dos atores envolvidos. A Tabela 4, que é semelhante à Tabela 2, página 38**,** mostra alguns elementos e associações encontradas nesse contexto.

Tabela 4: Alguns elementos e associações envolvidos em um cruzamento de vias.

Atores	Motorista, Pedestre
Objetos	Veículo, Semáforo, Placa de Sinalização, Faixa de Pedestre, Calçadas
Grupo	Pedestres, Veículos
Atividades	Identificar sinalização da via, Identificar cores do semáforo, obedecer, parar, prosseguir, aguardar, dar a preferência.
Tempo	Tempo do semáforo
Espaço	Vias de trânsito de veículos, Faixa de Pedestre, Calçadas
Abstrações	Veículos.transita.Vias Pedestre.transita.calçadas Pedestre.transita.pela_faixa_de_pedreste Vias.possui.placa_de_sinalização Vias.possui.semáforo Semáforo.possui.cores Veículo.possui.motorista Motorista.identifica.placa_de_sinalização Motorista.identifica.cores_do_semáforo Pedestre.identifica.placa_de_sinalização Pedestre.identifica.cores_do_semáforo

Na tabela apresentada os elementos são classificados conforme sua dimensão e também são mostradas as abstrações de algumas associações. Alguma associação pode se repetir, mas os parâmetros são diferentes, como por exemplo:

‘Veículos.transita.Vias’ e *‘Pedestres.transita.calçadas’*, onde a associação *‘transita’* faz parte dessas associações.

As figuras que serão apresentadas mostrarão as ontologias de alguns dos elementos apresentados na Tabela 4. A Figura 43 mostra a definição da ontologia Motorista. O campo ‘*Dimension*‘ indica a dimensão à qual a ontologia pertence. No campo *‘Comment’* é feita a definição conceitual da dimensão. Já o campo ‘*ImportOntology*’ faz menção à ontologia ‘*Veículo*’ definida pela própria linguagem *OntoL-Forum*, indicando que ela faz parte do mesmo contexto, e outros campos definem as condições necessárias e suficientes para sua existência.

```
Ontology-Class Motorista [motorista]
{
   Dimension: Actor
   Comment: 'Um motorista é o condutor de qualquer veículo de tração mecânica. Para se
            tornar um motorista uma pessoa tem que ter idade mínima exigida pela
            regulamentação local, e é submetida a curso, exame teórico e prático. Todo
            motorista na condução de um veículo numa via de trânsito exerce alguma
            atividade. '
   ImportOntology: Veículo
   SuperClass: Pessoa
   Property: Motorista.Nome
       Value: string
       MinCard: 1
       MaxCard: 1
   Property: Motorista.Idade
       Value: integer
       MinCard: 1
       MaxCard: 1
   Property: Motorista.Curso
       Value: string
       MaxCard: 1
   Property: Motorista.Exame_teórico
       Value: string
       MaxCard: 1
   Property: Motorista.Exame_prático
       Value: string
       MaxCard: 1
   Property: Motorista.Veículo
       ValueClass: (instanceOf Veículo)
       MinCard: 1
   Property: Motorista.Atividades
       Value: Atividades = {atividade}
       MinCard: 1
   ConstraintOntology: ( Motorista.Idade => 18
                         Motorista.Nome = Pessoa.Nome
                         Motorista.Idade = Pessoa.Idade  )
}
```

Figura 43 – Estudo de Caso - Ontologia Motorista em *OntoL-Forum* .

O campo ‘*SuperClass*’ indica que a ontologia Motorista é subclasse da ontologia ‘*Pessoa*’. Os atributos para esta classe são: nome, idade, curso, exame prático, exame teórico, veículo e atividades. O atributo *Motorista.Nome* significa que

todo motorista possui um nome, e somente um. O mesmo acontece para o atributo *Motorista.Idade*. Os atributos *Motorista.Curso, Motorista.Exame_teórico e Motorista.Exame_prático* significam que todo motorista tem que ser submetido a um curso e dois exames pelo menos uma vez na vida. O atributo *Motorista.Veículo* significa que um motorista conduz pelo menos um veículo, e que esse veículo pertence à classe Veículo. O atributo *Motorista.Atividades* significa que um motorista exerce uma atividade entre muitas na condução do veículo. Já o atributo *ConstraintOntology* faz menção às restrições, que neste caso, a idade tem que ser maior ou igual a 18 anos e que, o nome e a idade do motorista é de uma pessoa.

A Figura 44 mostra a definição da ontologia Pedestre. O campo '*Dimension*' indica a dimensão à qual a ontologia pertence. No campo *'Comment'* é feita a definição conceitual da dimensão. Já o campo '*ImportOntology*' faz menção a ontologia '*Via*' definida via linguagem *OntoL-Forum*, indicando que ela faz parte do mesmo contexto, e os outros campos definem as condições necessárias e suficientes para sua existência.

```
Ontology-Class Pedestre [pedestre]
{
   Dimension: Actor
   Comment: 'Pedestre é todo aquele que está a pé ou anda a pé. O pedestre tem um espaço
            reservado para sua circulação em uma via de trânsito e está sujeito a sua
            sinalização. O Pedestre em circulação por uma via de trânsito exerce alguma
            atividade.'
   ImportOntology: Via
   SuperClass: Pessoa
   Property: Pedestre.Nome
       Value: string
       MinCard: 1
       MaxCard: 1
   Property: Pedestre.Espaço
       ValueClass: (fieldOf Via)
       MinCard: 1
   Property: Pedestre.Atividades
       Value: Atividades = {atividade}
       MinCard: 1
   ConstraintOntology: (Pedestre.Nome = Pessoa.Nome)
}
```

Figura 44 – Estudo de Caso - Ontologia Pedestre em *OntoL-Forum* .

O campo '*SuperClass*' indica que a ontologia Pedestre é subclasse da ontologia '*Pessoa*'. Os seus atributos são: nome, espaço e atividades. O atributo *Pedestre.Nome* significa que todo pedestre possui um nome, e somente um. O atributo *Pedestre.Espaço* significa que todo pedestre tem um espaço reservado para sua circulação na via. O atributo *Pedestre.Atividades* significa que um pedestre

exerce uma atividade entre muitas em circulação pela via. Já o atributo *ConstraintOntology* faz menção a uma restrição que significa que o nome do pedestre é de uma pessoa.

A Figura 45 mostra a definição da ontologia Veículo. O campo '*Dimension*' indica a dimensão à qual a ontologia pertence. No campo *'Comment'* é feita a definição conceitual da dimensão. Já o campo '*ImportOntology*' faz menção às ontologias '*Via*' e '*Motorista*' definidas via linguagem *OntoL-Forum*, indicando que elas fazem parte do mesmo contexto, e os outros campos definem as condições necessárias e suficientes para sua existência. Os seus atributos são: condutor, capacidade e tipo.

O atributo *Veículo.Condutor* significa que todo veículo possui um condutor pelo menos, ou seja, que é conduzido por um motorista. O atributo *Veículo.Capacidade* significa que todo veículo tem uma capacidade. O atributo *Veículo.Tipo* significa que um veículo tem um tipo específico.

```
Ontology-Class Veículo [veículo]
{
   Dimension: Object
   Comment: 'Um veículo é qualquer meio para transportar ou conduzir pessoas, motorizado ou
            não. Os veículos transitam por vias conforme regulamentação de trânsito, por
            meio de um condutor e pode ser de diferentes tipos e de várias capacidades. '
   ImportOntology: Via, Motorista
   SuperClass:
   Property: Veículo.Condutor
       ValueClass: (instanceOf Motorista)
       MaxCard: 1
   Property: Veículo.Capacidade
       Value: integer
       MinCard: 1
   Property: Veículo.Tipo
       Value: string
       MinCard: 1
   ConstraintOntology: ()
}
```

Figura 45 – Estudo de Caso - Ontologia Veículo em *OntoL-Forum* .

A Figura 46 mostra a definição da ontologia Via. O campo '*Dimension*' indica a dimensão à qual a ontologia pertence. No campo *'Comment'* é feita a definição conceitual da dimensão. Já o campo '*ImportOntology*' faz menção as ontologias '*Pedestre*' e '*Veículo*' definidas via linguagem *OntoL-Forum*, indicando que elas fazem parte do mesmo contexto, e os outros campos definem as condições necessárias e suficientes para sua existência. Os seus atributos são: calçadas, faixa, regulamentação e sinalização.

Alguns dos atributos da ontologia Via aparecem no contexto da travessia de veículos como pertencendo a duas dimensões. Esses atributos são: calçadas e faixa de pedestre. Nas abstrações associativas '*Pedestre.transita.calçadas'* e '*Pedestre.transita.faixa_de_pedestre'*, ambos atributos são vistos como se pertencessem à dimensão Espaço, como um local por onde o pedestre transita. Na definição da ontologia Via esses atributos aparecem como objetos que compõem uma via, ou seja, seus valores são valores pertencentes à dimensão Objeto. Em outras palavras, a ontologia Via pertence à dimensão Espaço, mais alguns de seus atributos são considerados objetos.

Em *OntoL-Forum* se algum atributo não tem um valor especificado, ou seja, não é uma instância, nem conjunto, nem *string* ou *integer*, seu valor pode ser tipificado pela dimensão a que pertença através da palavra chave '*typeOf* '. Na definição da ontologia Via alguns atributos são especificados dessa maneira. Os atributos *Via.Calçada*, *Via.Faixa* e *Via.Sinalização* recebem o valor conforme sua dimensão, e a cardinalidade dos respectivos atributos significa que todos fazem parte da via. Já o atributo *Via.Regulamentação* significa que a via está sujeita a uma única regulamentação.

```
Ontology-Class Veículo [veículo]
{
   Dimension: Object
   Comment: 'Um veículo é qualquer meio para transportar ou conduzir pessoas, motorizado ou
            não. Os veículos transitam por vias conforme regulamentação de trânsito, por
            meio de um condutor e pode ser de diferentes tipos e de várias capacidades. '
   ImportOntology: Via, Motorista
   SuperClass:
   Property: Veículo.Condutor
       ValueClass: (instanceOf Motorista)
       MaxCard: 1
   Property: Veículo.Capacidade
       Value: integer
       MinCard: 1
   Property: Veículo.Tipo
       Value: string
       MinCard: 1
   ConstraintOntology: ()
}
```

Figura 46 – Estudo de Caso - Ontologia Via em *OntoL-Forum* .

A Figura 47 mostra a definição ontológica da atividade Identificar Sinalização. O campo '*Dimension*' indica a dimensão a qual a ontologia pertence. No campo *'Comment'* é feita a definição conceitual da dimensão. Já o campo '*ImportOntology*' faz menção às ontologias '*Via*', '*Pedestre*' e '*Motorista*', definidas na linguagem *OntoL-Forum* indicando que elas fazem parte do mesmo contexto, e os

outros campos definem as condições necessárias e suficientes para sua existência. Os seus atributos são: Pedestre e Motorista.

```
Ontology-Class Identificar Sinalização [identificar_sinalização]
{
   Dimension: Activity
   Comment: 'A atividade identificar sinalização é realizada por pedestres e motoristas.'
   ImportOntology: Via, Pedestre, Motorista
   SuperClass:
   Property: Identificar_sinalização.Pedestre
       ValueClass: (instanceOf Pedestre)
       MinCard: 1
   Property: Identificar_sinalização.Motorista
       ValueClass: (instanceOf Motorista)
       MinCard: 1
   ConstraintOntology: ()
}
```

Figura 47 – Estudo de Caso - Ontologia Identificar Sinalização em *OntoL-Forum* .

Os atributos dessa atividade Identificar Sinalização significam que ela envolve pelo menos um pedestre e um motorista conforme a cardinalidade mencionada, e que ambos são instâncias das classes Pedestre e Motorista respectivamente.

```
Ontology-Class Semáforo [semáforo]
{
   Dimension: Object
   Comment: 'O semáforo é um método de comunicação visual, que usa leds com cores
             diferentes usado para determinar quanto ao direito de passagem de pedestres e
             veículos em um cruzamento de vias.'
   ImportOntology: Via
   SuperClass:
   Property: Semáforo.Leds
       Value: Leds = {'green', 'red', 'yellow'}
       MinCard: 1
   ConstraintOntology: ()
}
```

Figura 48 – Estudo de Caso - Ontologia Semáforo em *OntoL-Forum* .

A Figura 48 mostra a definição da ontologia Semáforo. O campo '*Dimension*' indica a dimensão a qual a ontologia pertence. No campo *'Comment'* é feita a definição conceitual da dimensão. Já o campo '*ImportOntology*' faz menção à ontologia '*Via*' definida via linguagem *OntoL-Forum*, indicando que ela faz parte do mesmo contexto. Neste caso, a ontologia possui um único atributo: o atributo *Semáforo.Leds*, que tem como valor um conjunto de valores pré-definidos determinados pela palavra-chave '={ }', e sua cardinalidade significa que um semáforo possui pelo menos um conjunto de leds.

```
Ontology-Class Tempo Semáforo [tempo_semáforo]
{
   Dimension: Intervalo
   Comment: 'O tempo do semáforo é dividido em três tempos através da mudança de cores de
            seus leds, que ocorre em um determinado momento. '
   ImportOntology: Semáforo
   SuperClass:
   Property: Tempo_semáforo.Mudança
       Value: string
       MinCard: 1
   ConstraintOntology: ()
}
```

Figura 49 – Estudo de Caso - Ontologia Tempo do Semáforo em *OntoL-Forum* .

A Figura 49 mostra a definição da ontologia Tempo Semáforo. O campo '*Dimension*' indica a dimensão a qual a ontologia pertence. No campo *'Comment'* é feita a definição conceitual da dimensão. Já o campo '*ImportOntology*' faz menção a ontologia '*Semáforo*' definida via linguagem *OntoL-Forum*, indicando que ela faz parte do mesmo contexto. Neste caso, a ontologia também possui um único atributo. O atributo *Tempo_semáforo.Mudança* significa que o tempo do semáforo é feito por uma mudança de cores.

```
Ontology-Association Transita [pedestre faixa_pedestre]
{
   Abstration: Utilization
   Comment: 'O pedestre tem por obrigação atravessar uma via pela faixa de pedestre. '
   ImportOntology: Pedestre, Via
   Superclasse:
   Property: Transita.Pedestre
       ValueClass: (intanceOf Pedestre)
       MinCard: 1
   Property: Transita.Faixa_pedestre
       ValueClass: (propertyOf Via)
       MinCard: 1
       MaxCard: 1
   ConstraintOntology:
}
```

Figura 50 – Estudo de Caso - Associação em *OntoL-Forum* .

A Figura 50 mostra a definição de uma ontologia para a associação Transita, exemplificada na Tabela 4, página 54. A ontologia que define esta associação tem como paramentos os argumentos *pedestre* e *faixa_pedestre*. O campo '*Abstraction'* é tipificado como uma associação de utilização (utilization). No campo *'Comment'* é feita a definição conceitual da associação, e os outros campos definem as condições necessárias e suficientes para sua existência. Já o campo '*ImportOntology*' faz menção as ontologias '*Pedestre*' e '*Via*' definidas via linguagem *OntoL-Forum*, indicando que elas fazem parte do mesmo contexto. Os atributos da associação são: pedestre e faixa_pedestre.

Neste caso, o atributo *Transita.Pedestre* é uma instância da classe Pedestre, e sua cardinalidade mostra que esta associação existe para no mínimo um pedestre. Já o atributo *Transita.Faixa_pedestre* é uma propriedade da classe Via determinada pela palavra-chave '*propertyOf*', e sua cardinalidade mostra que esta associação existe para uma faixa de pedestre. A palavra-chave '*propertyOf*' é usada em *OntoL-Forum* quando se associa ou faz menção a um único atributo de uma ontologia, diferente do '*fieldOf*', que é mencionado para mais de um valor.

A Figura 51 mostra a definição ontológica para uma associação Identifica, exemplificada na Tabela 4, página 54. A ontologia que define esta associação tem como parâmetros os argumentos *motorista* e *cores_semáforo*. O campo '*Abstration'* é tipificado como uma associação genérica (Generic). No campo *'Comment'* é feita a definição conceitual da associação, e os outros campos definem as condições necessárias e suficientes para sua existência. Já o campo '*ImportOntology*' faz menção as ontologias '*Semáforo*', '*Motorista*' e '*Via*' definidas via linguagem *OntoL-Forum*, indicando que elas fazem parte do mesmo contexto. Os atributos da associação são: motorista e cores_semáforo.

```
Ontology-Association Identifica [motorista cores_semáforo]
{
   Abstration: Generic
   Comment: 'O motorista tem por obrigação observar as cores do semáforo. '
   ImportOntology: Motorista, Semáforo, Via
   Superclass:
   Property: Identifica.Motorista
        ValueClass: (intanceOf Motorista)
        MinCard: 1
   Property: Identifica.Cores_semáforo
        ValueClass: (propertyOf Semáforo)
        MinCard: 1
        MaxCard: 1
   ConstraintOntology:
}
```

Figura 51 – Estudo de Caso - Associação em *OntoL-Forum* .

Neste caso, o atributo *Identifica.Motorista* é uma instância da classe Motorista, e sua cardinalidade mostra que esta associação existe para no mínimo um motorista. Já o atributo *Identifica.Cores_semáforo* é uma propriedade da classe Semáforo, e sua cardinalidade mostra que esta associação existe para uma faixa de semáforo.

4.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentadas a semântica e sintaxe da linguagem *OntoL-Forum*, destacando-se sua finalidade. As ontologias aqui formalizadas mostram a aplicabilidade da linguagem sobre as regras especificadas pela linguagem *L-Forum*, no Capítulo 3.

Foi apresentada a ontologia de cada dimensão especificada pelo modelo *M-Forum,* e um estudo de caso baseado nas abstrações mostradas na Tabela 4, página 54, definindo ontologias e associações dos elementos envolvidos na abstração.

CAPÍTULO 5

CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

O uso de ontologia passou a ser um método muito usado para representar um conjunto de conceitos dentro de um domínio, e hoje já existem várias linguagens para esse propósito, mas não uma específica para modelagem de regras em ambientes interativos.

O modelo *M-Forum* proporciona uma semântica que permite separar todos os elementos de uma interação em uma determinada dimensão, e suas associações, mas não permitia a especificação ontológica de suas dimensões. A falta de tal requisito incentivou o início deste trabalho na busca por uma solução que atendesse a necessidade desse modelo, surgindo a idéia de criar uma extensão a sua linguagem de especificação de regras, a *L-Forum*, que possibilitasse a especificação ontológica de suas dimensões e associações. Dentre muitas linguagens, a linguagem para descrição de ontologia, a *Ontolingua*, foi utilizada como base para essa adaptação, já que possui uma igualdade de semântica para especificar classes e relacionamentos.

Com a extensão da linguagem *L-Forum,* foi desenvolvida a linguagem de denominação *OntoL-Forum*, que vem de uma junção da semântica da linguagem *Ontolingua*, com a estrutura da linguagem *L-Forum*, mas é independente para a definição das ontologias. No estudo de caso foi mostrada a sua aplicabilidade através das definições ontológicas das dimensões e associações abstraídas com base no modelo *M-Forum*, concretizando sua finalidade.

Contudo, o modelo de especificação de regras *M-Forum* ficou mais expressivo e enriquecido com esta nova adaptação, que possibilita a modelagem ontológica de suas dimensões e associações.

5.2 CONTRIBUIÇÃO DA PESQUISA

Este trabalho cria uma linguagem que servirá como ferramenta de apoio a projetistas que desenvolvem projetos na área de especificação em regras em ambientes interativos. Com a modelagem dos elementos envolvidos nas dimensões e suas associações, aumenta a expressão dos projetos desenvolvidos através do modelo de especificação de regras *M-Forum*, facilitando o entendimento e a compreensão.

O trabalho atende também a necessidade de pesquisadores que procuram uma ferramenta para essa finalidade em específico, já que a maioria das linguagens de definição de ontologias está voltada para *Web*. Por esse motivo, este trabalho tem contribuído não somente para a melhoria do modelo de especificação de regras *M-Forum*, como também para a comunidade da área de Ciência da Computação.

5.3 TRABALHOS FUTUROS

Futuros trabalhos podem aumentar a expressividade da linguagem, como por exemplo, uma tradução para linguagem *OWL,* que é uma linguagem para escrita de ontologia específica para *Web.* Outro trabalho a ser desenvolvido é permitir que um editor de regras em *L-Forum,* que está em desenvolvimento, possa ser estendido para a autoria de ontologias em *OntoL-Forum,* baseando-se nos parâmetros estabelecidos na definição das regras.

Outro trabalho a ser realizado é a criação de um método verificador de ontologias, que possa aumentar a consistência das ontologias definidas pela linguagem *OntoL-Forum*, no desenvolvimento de projetos.

REFERÊNCIAS BIBLIOGRÁFICAS

BOLEY, H. et al.: **Rule Interchange on the Web**, Lecture Notes in Computer Science - LNCS, Springer-Verlag, Berlin Heidelberg, vol. 4636, G. Antoniou et al. (Eds.): Reasoning Web 2007, pp. 269–309, 2007.

BECHHOFER, S. **Ontology Language Standardisation Efforts,** OntoWeb: Ontology-based Information Exchange for Knowledge Management and Electronic Commerce - Information Management Group Department of Computer Science University of Manchester - IST Project IST-2000-29243.

BRAY T., PAOLI J., SPERBERG-McQUEEN C. M. **Markup Language (XML) 1.0**. W3C Recommendations, 10 February 1998. Relatório Técnico. Disponível em: <http://www.w3.org/TR/1998/REC-xml-19980210>. Acesso em: 20 de Fevereiro de 2008.

BRESSAN, R. L. **Verificação de Conflitos em Regras de Interação em Ambientes Colaborativos**. 2008. 76 f. Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba, Piracicaba.

BRUIJN, J.:**RIF RDF and OWL Compatibility,** W3C Working Draft 30 July 2008, Disponível em: < http://www.w3.org/TR/2008/WD-rif-rdf-owl-20080730/ > Acesso em: 20 de Fevereiro de 2008.

CAMOLESI Jr., L.; MARTINS, L. E. G. **Um Modelo de Interações para Definição de Regras de Jogos**. In: II Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital (SBGAMES), 2005, São Paulo (SP) - Brasil. Anais do IV Workshop Brasileiro de Jogos e Entretenimento Digital, II Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital. São Paulo (SP) - Brasil:

Universidade de São Paulo (USP). Sociedade Brasileira de Computação (SBC), 2005. v. 1. p. 162-173.

DING, Z.; PENG, Y.: **A probabilistic extension to ontology language OWL,** System Sciences, 2004. Proceedings of the 37th Annual Hawaii International, IEEE Conference on 5-8 Jan. 2004.

GOMEZ-PEREZ, A. **Tutorial on Ontological Engineering**. Internacional Joint Conference on Artificial Intelligence – IJCAI'1999. Estocolmo, Suécia. <http://www.ontology.org/main/papers/madrid-tutorials.htm>, Acesso em: 30 de Março de 2008.

GRUBER, T. R. **Ontolingua: A Mechanism to Support Portable Ontologies**. Stanford University, Knowledge Systems Laboratory, Technical Report KSL 91-66, 1992. Disponível em: <ftp://ksl.stanford.edu/pub/KSL_Reports/KSL-91-66.ps.gz> Acesso em: 30 de Março de 2008.

GRUBER, T. R. **A Translation Approach to Portable Ontology Specifications**. Journal Appeared in Knowledge Acquisition, volume 5(2):199-220, 1993. Disponível em: <http://tomgruber.org/writing/ontolingua-kaj-1993.pdf > Acesso em: 30 de Março de 2008.

GUIMARÃES, F. J. Z.; LUCENA, C. J. P.: **Ontologies use in B2C domain**, 2002. 195p. Dissertação (Mestrado em Informática) - Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

MARTINS, L. E. G.; CAMOLESI Jr., L.:**FORUM: Modelo e Linguagem para Especificação de Regras em Ambientes Colaborativos**, RITA – Revista de Informática Teórica e Aplicada Vol. 15, nº2, pag. 9 -26, 2008.

McGUINNESS, D. L.; HARMELEN, F. V.:**OWL Web Ontology Language Overview**, W3C Recommendation 10 February 2004, Disponível em: < http://www.w3.org/TR/2004/REC-owl-features-20040210/>, Acesso em: 12 de Fevereiro de 2008.

NOY, N. F.; McGUINNESS, D. L.: **Ontology Development 101: A Guide to Creating your First Ontology**, 2001, Stanford Medical Informatics Technical Report SMI, Disponível em:< http://www-ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf >, Acesso em: 30 de Março de 2008.

PASCHKE, A. et al.:**RIF Use Cases and Requirements,** W3C Editor's Draft 30 July 2008, Disponível em:< http://www.w3.org/2005/rules/wg/draft/ED-rif-ucr-20080730/ > Acesso em: 20 de Fevereiro de 2008.

RIF WORKING GROUP**, FAQ to Rule Interchange Format (RIF)**. Disponível em:< http://www.w3.org/2005/rules/wiki/RIF_FAQ>, Acesso em: 20 de Fevereiro de 2008.

SEMPREBOM, T.; CAMADA M. Y.; MENDONÇA I.: **Ontologias e Protégé**, Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina (UFSC) - Florianópolis, SC, Brasil, 2007. Disponível em: < http://www.das.ufsc.br/~gb/pg-ia/Protege07/ontologias.pdf >, Acesso em: 4 de Fevereiro de 2008.

SMITH, M. K.; WELTY, C.; McGUINNESS, D. L.: **OWL Web Ontology Language Guide,** W3C Recommendation - 10 February 2004, Disponível em:<http://www.w3.org/TR/2004/REC-owl-guide-20040210/ >, Acesso em: 2 de Fevereiro de 2008.

USEHOLD, M.; GRUNINGER, M. **Ontologies: Principles, methods and applications**. Cambridge Journal, Knowledge Engineering Review, volume 11, Page(s): 93 – 136, 1996.

ZHIHONG, Z.; MINGTIAN, Z.; **Web Ontology Language OWL and its description logic foundation,** Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on 27- 29 August 2003, Page(s):157 – 160.

APÊNDICE

ONTOL-FORUM

<ontology-class> ::=	‘Ontology-Class’ <class name> ‘[‘ <instance> ’ | ‘abstract’]’ ‘{’ <context class> ‘}’
<ontology-association> ::=	‘Ontology-Association’ < association name> ‘[‘ <argumentos> ’]’ ‘{’ <context association> ‘}’
<context class> ::=	‘Dimension:’ <dimension> ‘Comment:’ <comment> ‘ImportOntology:’ <importOntology> ‘SuperClass:’ <superclass> ‘Property:’ <property> ‘ConstraitOntology: (’ <constraintOntology> ‘)’
<context Association > ::=	‘Abstraction:’ <abstraction> ‘Comment:’ <comment> ‘ImportOntology:’ <importOntology> ‘SuperClass:’ <superclass> ‘Property:’ <property> ‘ConstraitOntology: (’ <constraintOntology> ‘)’
<instance> ::=	<string>
<argumentos> ::=	<string>
<dimension> ::=	‘actor’ | ‘object’ | ‘space’ | ‘time’ | ‘activity’
<abstraction> ::=	‘Property’ | ‘Utilization’ | ‘Grouping’ | ‘Classification’ | ‘Generalization’ | ‘Composition’ | ‘Generic’
<comment> ::=	<string>
<importOntology> ::=	<ontology name> | <url>
<superclass> ::=	< ontology name>
<property> ::=	<string> <constraint>
<constraintOntology> ::=	<string> | <sentence>
<class name> ::=	<string>
<association name> ::=	<string>
<constraint> ::=	‘ValueClass:’ ‘(‘ <valueclass> ‘)’ ‘MinCard:’ <cardinality minima> ‘MaxCard:’ <cardinality maxima>
<sentence> ::=	‘(‘<string> <operador> <string>’)’
<operador> ::=	‘<’ | ‘>’ | ‘=’ | ‘<=’ | ’>=’ | ‘≠’
<valueclass> ::=	‘fieldOf’ <string> | ‘instanceOf’ <string> | ‘tipeOf’ <string> | ‘propertyOf’ <string>
<url> ::=	<string>
<string> ::=	<letter>, {<letter>}
<cardinality minima> ::=	<integer value>
<cardinality maxima>::=	<integer value>
<integer value> ::=	<digit>, {<digit>}
<letter> ::=	´a´ | ´b´ | ´c´ | ´d´ | ´e´ | ´f´ | ´g´ | ´h´ | ´I´ | ´j´ | ´k´ | ´l´ | ´m´ | ´n´ | ´o´ | ´p´ | ´q´ | ´r´ | ´s´ | ´t´ | ´u´ | ´v´ | ´x´ | ´w´ | ´y´ | ´z´
<digit> ::=	´1´ | ´2´ | ´3´ | ´4´ | ´5´ |´6´ | ´7´ | ´8´ | ´9´ | ´0´

ANEXO

L-Forum

<rule> ::=	´Rule´ <rule name> ´[´<status>´]´ ´{´ [<context>] ´Body ::´<definition> [<regime>] ´}´
<context>::=	[´Parameters: (´<parameters>´)´] [<applicability>]
<definition> ::=	<condition> | <action> | <rule call> [<definition>]
<regime> ::=	<survivability> [´Priorities:´ <priority>]
<parameters> ::=	<identifier>´:´<element> [´,´ <parameters>]
<element> ::=	<actor> | <group> | <object> | <space> | <time> | <association> | <activity> | <operation>
<type> ::=	´actor´ | ´group´ | ´object´ | ´space´ | ´time´ | ´association´ | ´activity´ | ´operation´
<applicability> ::=	´Applicability::´ <condition expression>
<survivability> ::=	´Survivability::´ <condition expression>
<condition> ::=	´If ´ <condition expression> ´then {´ <definition> ´}´ [´else {´ <definition>´}´]
<action> ::=	´Action: (´ <supreme action> | <definition action> | <attribution action> | (<actor> <normative operator> {<activity> (<actor> | <object>)} [<space attribution operation> <space>] [<time attribution operation> <time>]) [´);´ <action>] ´);´
<supreme action> ::=	<actor> <normative operator> <primitive operator> (<element>|<domain>|(´is part of´|´is a´) <element>) | <actor> <normative operator> <primitive group operator> <group> <element>)
<definition action> ::=	<actor> ´set´ <status>
<attribution action> ::=	<actor> ´attribute´ (<value>|<formula>| ((next | prior) (<value domain>|<domain name>)) <attribute>
<condition expression> ::=	´(´ (<attribute><attribute condition operator>(<value>| ([´all´|´any´] (<value domain>|<domain name>)) | ((<element>|<rule name>) <status condition operator> <value>) | (<element>) <element group operator> <group>) | (<element>) <group group operator> <group>) | ([not] ´exist´<association>) | (<element> <space condition operator> (<space> | (<group>|<domain name>))) | (<attribute><space condition operator>(<value>|´here´|([´all´|´any´] (<value domain>|<domain name>)))) | (<attribute><time condition operator>(<value>|´now´|([´all´|´any´] (<value domain>|<domain name>)))) | (<actor> <activity condition operator> <activity>) | (<condition expression> (and | or)) ´)´
<rule call> ::=	´Rule (´ <rule name> ´ (´<parameters>´)´ <normative operator> [´);´ <rule call>] ´);´
<priority> ::=	<name> [´,´ <priority>]
<group> ::=	[(<identifier>) |´any´ | ´all´| <integer value>] [<association>] [´:´<name>] [´:Group´]
<actor> ::=	[(<identifier>) |´any´ | ´all´| <integer value>] [<association>] [<group>´.´] [´:´<name>] [´:Actor´]
<activity> ::=	[<activity>´.´] <name> [´:Activity´]
<operation> ::=	<activity>´.´<name> [´:Operation´]
<object> ::=	[(<identifier>) |´any´ | ´all´| <integer value>] [<association>] [<object>´.´] [´:´<name>] [´:Object´]
<space> ::=	[(<identifier>) |´any´ | ´all´| <integer value>] [<association>] [<space>´.´] [´:´<name>] [´:Space´]
<time> ::=	[(<identifier>) |´any´ | ´all´| <integer value>] [<association>] [´:´<name>] [´:Time´]
<association> ::=	<element>´.´<name> [´.´<association>] [´:Association´]
<attribute> ::=	<element>´.´<name> [´:Attribute´]
<domain> ::=	<name> (<value domain> | <grouping>)
<value domain> ::=	´(´ (<numeric value> {´,´ <numeric value>}) | (<string> {´,´ <string>}) ´)´
<grouping> ::=	(<type> <name> <attribute condition operator> (<value>|([´all´| ´any´](<value domain>|<domain name>))) {(and| or) <grouping>}) | (<element> {´,´ <element>})
<status> ::=	<element> <status attribution operator> <value>
<primitive group operator> ::=	´insert´ | ´delete´ | ´update´
<primitive operator> ::=	´create´ | ´destroy´
<group element operator> ::=	´∈´ | ´∉´ |
<group group operator>	´⊆´ | ´⊄´ | ´⊂´

::=	
<activity condition operator> ::=	[´not´] ´has´
<normative operator> ::=	´right´ | ´prohibition´ | ´obligation´ | ´dispensation´
<activity attribution operator> ::=	´receive´
<status attribution operator> ::=	´put on´ | ´move to´
<space attribution operator> ::=	´==´ | ´inside´ | ´outside´ | ´north´ | ´south´ | ´east´ | ´west´
<time attribution operator> ::=	´in´ | ´on´ | ´at´
<attribute condition operator> ::=	´<´ | ´<=´ | ´>´ | ´>=´ | ´=´ | ´<>´
<time condition operator> ::=	´<´ | ´<=´ | ´>´ | ´>=´ | ´=´ | ´<>´ | ´precedes´ | ´succeeds´ | ´directly precedes´ | ´directly succeeds´| ´overlaps´| ´is overlapped by´| ´occurs during´| ´contains´| ´starts´ | ´is started with´| ´finishes´| ´is finished with´| ´coincides with´
<space condition operator> ::=	´<´ | ´<=´ | ´>´ | ´>=´ | ´=´ | ´<>´ | ▪´not equal´ | ´inside´ | ´outside´ | ´intersect´ | ´meet´ | ´overlap´ | ´north´ | ´south´ | ´east´ | ´west´
<arithmetic operator> ::=	´+´ | ´-´ | ´/´ | ´*´
<status condition operator> ::=	[´not´] ´is´
<rule name> ::=	<string>
<name> ::=	<string>
<domain name>::=	<string>
<string> ::=	<letter>,{<letter>}
<integer value> ::=	<digit>,{<digit>}
<real value> ::=	<digit>,{<digit>},´.´,<digit>,{<digit>}
<numeric value> ::=	<integer value> | <real value>
<digit> ::=	´1´ | ´2´ | ´3´ | ´4´ | ´5´ |´6´ | ´7´ | ´8´ | ´9´ | ´0´
<letter> ::=	´a´ | ´b´ | ´c´ | ´d´ | ´e´ | ´f´ | ´g´ | ´h´ | ´I´ | ´j´ | ´k´ | ´l´ | ´m´ | ´n´ | ´o´ | ´p´ | ´q´ | ´r´ | ´s´ | ´t´ | ´u´ | ´v´ | ´x´ | ´w´ | ´y´ | ´z´
<identifier> ::=	<letter>|<identifier> [,<letter> | <integer value>]
<formula>::=	´(´(<attribute>|<numeric value>|<formula>) [<arithmetic operator> <formula> ´)´
<value>::=	<string> | <numeric value>