Método e Ferramenta para Ensino de Projeto e Desenvolvimento de Ontologias

João Carlos Gluz

Pós-Graduação em Computação Aplicada (PIPCA) – Unisinos – RS – Brazil

jcgluz@unisinos.br

Abstract. *This paper presents a teaching method for the design and development of ontologies and a tool created to support this method. The teaching method covers theoretical and practical aspects, combining Description Logics and formal methods contents, with practice in the form of a development project on an OWL ontology, which is based on current Ontology Engineering methodologies. The tool developed to support this method provides a learning environment for programming applications based on OWL ontologies, but it takes into account the logical and declarative aspects of ontologies. After the presentation of the method and the tool, a scenario with the application of the teaching method is introduced.*

Resumo. *Este trabalho apresenta um método de ensino para projeto e desenvolvimento de ontologias e uma ferramenta criada para apoiar o método. O método de ensino cobre aspectos teóricos e práticos, combinando conteúdos sobre Lógica Descritiva e métodos formais, com a prática na forma de projeto de ontologia OWL baseado nas metodologias atuais de Engenharia de Ontologias. A ferramenta desenvolvida para apoiar esse método provê um ambiente de ensino para programação de aplicações baseadas em ontologias OWL, mas que leva em conta os aspectos lógicos e declarativos dessas ontologias. Após a apresentação do método e da ferramenta, o trabalho mostra um cenário com a aplicação do método de ensino.*

1. Introdução

As tecnologias derivadas das ontologias e da web semântica têm condições de se tornar o novo paradigma de representação de conhecimentos, informações e dados, complementando ou, até mesmo, substituindo o atual paradigma baseado em bancos de dados relacionais. Essas tecnologias disponibilizam ferramentas como bancos de dados semânticos e bases de conhecimentos computacionalmente eficientes, além de permitir a especificação rigorosa e precisa do significado das informações compartilhadas entre softwares e sistemas, o que viabiliza todo um novo conjunto de aplicações.

Existem diversas áreas onde tecnologias baseadas em ontologias e na web semântica têm adquirido uma influência e atuação crescente. Na área Biomédica, por exemplo, há um considerável *corpus* de ontologias estabelecidas para os mais diversos domínios dessa área [BioPortal, 2016]. Na área de Informática na Educação também há um considerável e crescente interesse no uso dessas tecnologias. Apesar de ser uma tendência relativamente recente [Dicheva, 2009, Isotani, 2008], ontologias têm se mostrado úteis na concepção e construção de vários tipos de ambientes e sistemas educacionais, incluindo, entre outros, ambientes educacionais web [Silva, 2009; Bittencourt, 2009; Melis, 2009; Dzbor, 2009], modelos educacionais formais [Hayashi, 2009], gestão de conteúdos educacionais [Balog-Crisan, 2008; Kholief, 2012; Pahl, 2013] e gerência de objetos de aprendizagem [Porto, 2007; Carrion, 2007; Gluz, 2010].

DOI: 10.5753/cbie.sbie.2016.1066

Como consequência, habilidades de projetar e desenvolver ontologias e aplicações capazes de utilizar este novo tipo de tecnologia se tornam cada vez mais necessárias. Nesse contexto, surge uma questão importante:

> *Como ensinar as pessoas a desenvolverem as competências e habilidades para se tornarem engenheiros de ontologias.*

As metodologias de Engenharia de Ontologias [Gomez-Perez, 2004; Simperl, 2006; Sure, 2006; Grimm, 2011] assumem a existência deste papel, também denominado de "Ontologista" ou de "Engenheiro de Conhecimentos". Porém, em termos práticos hoje em dia simplesmente não existe a figura ou papel de "Engenheiro de Ontologias" (ou "Ontologista") no contexto de projetos de software, com o correspondente perfil de habilidades, competências e conhecimentos necessários para exercer esse papel. Assim é de fundamental importância buscar formas e métodos de ensino que visem sanar essa deficiência.

Independente disso, ainda existem outras questões mais profundas relacionadas a forma como se pode programar aplicações baseadas em ontologias, particularmente no caso de ontologias no padrão OWL [W3C, 2001]. A linguagem OWL é uma síntese de vários anos de pesquisa em representação de conhecimentos por ontologias. Essa linguagem trouxe inúmeras vantagens para as pesquisas sobre projetos e sistemas baseados em ontologias, oferecendo uma forma precisa, rigorosa e de tratabilidade computacional conhecida para a representação de conhecimentos ontológicos.

Entretanto, problemas e dificuldades importantes podem ser identificados com as abordagens existentes atualmente para programar ontologias OWL em aplicações. Muitas dessas dificuldades se originam no fato de OWL ser uma linguagem Lógica e Declarativa, ou seja, OWL é apenas uma sintaxe em formato XML/RDF para uma Lógica Descritiva formal. Uma ontologia OWL nada mais é que um modelo ou teoria lógico-formal de um determinado domínio. Manipular teorias e modelos lógico-formais em aplicações de software não é um processo trivial. Assim há uma tendência em se abstrair o fato de OWL ser uma Lógica Descritiva expressa em sintaxe RDF/XML quando se está desenvolvendo de uma dada aplicação. Porém, quando se faz essa abstração, sobra apenas o acesso e manipulação algorítmica aos elementos estruturais de OWL, ou seja, operações sobre uma estrutura de dados complexa baseada em grafos RDF, que é capaz de descrever textualmente entidades como "classes", "objetos" ou "relações", mas que não representa a semântica dessas entidades. O problema é que justamente a capacidade de tratar a semântica de classes e relações (de uma forma rigorosa, precisa e tratável) é que dá a OWL seu "poder de fogo" para ser o novo paradigma de representação de conhecimentos e informações (sem isso, a utilidade de OWL é até mesmo discutível).

Tendo essas questões em consideração, o presente trabalho apresenta os resultados de mais de dois anos ministrando uma disciplina de Pós-Graduação centrada no projeto e desenvolvimento formal de ontologias. Os resultados deste trabalho são compostos de um método de ensino para projeto e desenvolvimento de ontologias e de ferramentas para apoiar esse método. O método de ensino cobre aspectos teóricos e práticos, combinando conteúdos sobre Lógica Descritiva e métodos formais, com a prática na forma de projeto concreto de ontologia OWL baseado nas metodologias atuais de Engenharia de Ontologias. Uma ferramenta em particular, denominada de PrOWLog (*http://obaa.unisinos.br/prowlog*), foi desenvolvida para apoiar esse método. Essa ferramenta foi concebida como um ambiente de ensino para programação de aplicações baseadas em ontologias OWL que leva em conta os aspectos lógicos e declarativos das ontologias. PrOWLog oferece um ambiente de programação declarativo, abstrato e de alto nível que permite verificar e revisar modelos formais especificados em OWL. PrOWLog também é totalmente compatível com a ferramenta *Protege*, permitindo o uso integrado de ambas ferramentas para fins de ensino.

2. Metodologia de Ensino

O método de ensino para projeto e desenvolvimento de ontologias buscou aplicar as técnicas mais recentes de Engenharia de Ontologias [Gomez-Perez, 2004; Simperl, 2006; Sure, 2006; Grimm, 2011] em um contexto mais formal, levando em conta as propriedades, peculiaridades e dificuldades de se conceber, formalizar e avaliar ontologias com a linguagem OWL. Como não poderia deixar de ser, o ensino do projeto e desenvolvimento de ontologias segue uma metodologia de ensino orientada à projetos, onde cada aluno define em conjunto com o professor e após o processo inicial de apresentações, discussões e análises, qual tema (domínio de aplicação) e objetivo específico (aplicação) pretende trabalhar na disciplina.

A preocupação com a Engenharia de Ontologias, permitiu escapar um pouco do tópicos puramente formais relacionados às ontologias OWL e antever como essas ontologias podem se relacionar com os demais componentes de um sistema ou aplicação baseado em ontologias. Outro aspecto importante do método de ensino é a utilização sistemática e integrada de ferramentas computacionais nas diversas etapas e fases do projeto. Isso permitiu, sem perda de generalidade, diminuir um pouco da carga cognitiva de ensino de aspectos lógico-formais das ontologias,. O uso de ferramentas tornou este processo um pouco mais fácil. É importante ressaltar que foi necessário desenvolver uma ferramenta própria (o PrOWLog, descrito na Seção 3) para que o suporte de ferramentas fosse razoavelmente completo.

Os conhecimentos, habilidades e competências necessários a uma metodologia de projeto e implementação de ontologias são os principais conteúdos de ensino da disciplina. A Figura 1 apresenta uma síntese da metodologia de projeto de ontologias que foi empregada como base curricular da disciplina.

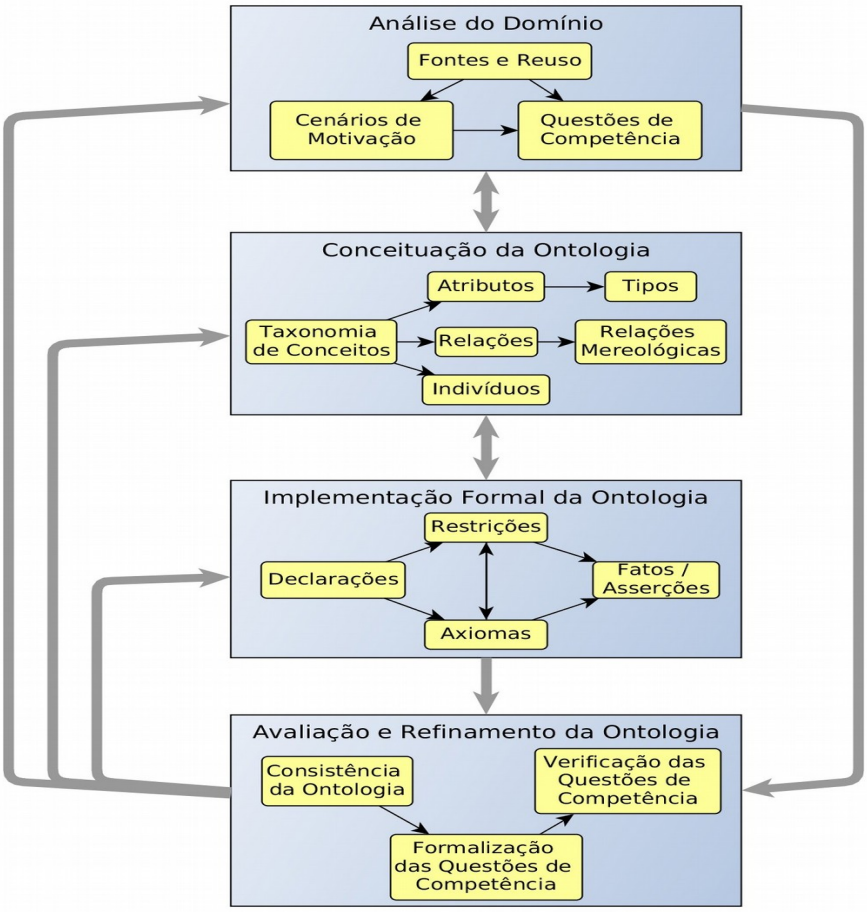

Figura 1. Metodologia e base curricular para ensino de ontologias

Esta é uma nova metodologia proposta neste trabalho e que foi concebida especificamente para fins de ensino. A metodologia integra os requisitos da metodologia formal de Gruniger e Fox [Gomez-Perez, 2004], referenciados principalmente na etapa inicial de Análise do Domínio e na etapa final de Avaliação e Refinamento da Ontologia, com os processos mais orientados para a modelagem e implementação da ontologia definidos na metodologia *Methontology* [Gomez-Perez, 2004] e referenciados nas etapas de Conceituação e Implementação Formal da Ontologia.

Seguindo a metodologia de Gruniger e Fox [Gomez-Perez, 2004], os *cenários de motivação* descrevem aquelas situações que serão enfrentadas pelas aplicações que utilizarão a ontologia a ser construída, enquanto que as *questões de competência*, escritas informalmente em linguagem natural, serão utilizadas para avaliar se a ontologia será capaz de satisfazer os requisitos dos cenários de motivação propostos. O reconhecimento de quais são as melhores *fontes de conhecimento* sobre o domínio aliada à pesquisa sobre possíveis *ontologias pré existentes* sobre este domínio e que poderiam ser reusadas, são processos que antecedem, mas que operam concomitantemente com a especificação dos cenários de motivação e das questões de competência. A Figura 2 mostra os métodos aplicados nesta etapa. São métodos voltados à internalização de conceitos, que envolvem a apresentação das formas de análise ontológica de domínios, seguida da discussão e da construção, pelos alunos, de sínteses textuais dos resultados desta etapa.

No contexto da *Methontology* [Gomez-Perez, 2004], a etapa de *conceituação da ontologia* ("*conceptualization*") leva a criação de um modelo conceitual para a ontologia. Este modelo é composto pela identificação de quais são os conceitos do domínio, devidamente organizados em uma taxonomia de classificação, seguida da definição de quais atributos ("*data properties*" na terminologia de OWL) e relações ("*object properties*" em OWL) estão vinculados aos conceitos. As relações do tipo parte/todo são definidas nesta etapa, criando o modelo "*mereológico"* do domínio (modelo de agregações ou de composições). Tipos de dados, especialmente se forem particulares ao domínio, também devem ser definidos nesta etapa. Em termos de método de ensino, esta etapa avança para o acompanhamento do projeto do aluno, com o apoio de ferramentas computacionais (no caso o ambiente *Protege*) para auxiliar na criação do modelo conceitual (ver Figura 2). O *Protege* foi escolhido não apenas pela sua ampla aceitação e utilização na comunidade de pesquisa, mas por permitir trabalhar com modelos gráficos e semi formais da ontologia, possibilitando um desenvolvimento evolutivo e em espiral do modelo ontológico.

A etapa de *implementação formal da ontologia* corresponde à transformação do modelo conceitual informal ou semi formal em um modelo lógico preciso, capaz de especificar formalmente cada um dos elementos do modelo conceitual (ver Figura 1). A semântica de OWL é baseada nas Lógicas Descritivas [Baader, 2003], assim cada uma das classes, relações e atributos descritos nessa linguagem correspondem à declarações, axiomas, restrições e asserções lógicas específicas. Nesta etapa, os aspectos lógicos de OWL são explorados e analisados de forma sistemática, para que o aluno possa ter uma compreensão da dimensão lógica e formal de qualquer ontologia nesta linguagem. Diferente das etapas anteriores esta etapa detalha a estrutura formal das Lógicas Descritivas, mostrando a sintaxe, semântica e mecanismos de inferência dessas Lógicas. Nesta etapa se adota com força completa a ferramenta PrOWLog que permite acompanhar o projeto e trabalhar interativamente com o modelo formal da ontologia (ver Figura 2). PrOWLog suporta uma notação compacta, baseada na sintaxe Manchester para OWL [W3C], para trabalhar com para os axiomas, declarações e asserções e OWL. PrOWLog pode trabalhar integrado aos motores de inferência *Pellet*, *Fact++* e *Hermit*, que são os mesmos disponíveis no *Protege*.

A etapa final de *avaliação e refinamento da ontologia* passa pela verificação da consistência dos axiomas da ontologia, combinada com a formalização e verificação das

questões de competência definidas na etapa inicial (ver Figura 1). Os axiomas e asserções formais definidos na etapa anterior entram como condições necessárias e suficientes para formalizar e verificar as questões de competência. O processo de revisão e refinamento da ontologia começa com as inconsistências e falhas encontradas neste processo de verificação formal. O uso dos motores de inferência integrados tanto ao PrOWLog quanto ao *Protege* se tornam fundamentais nesta etapa, permitindo a verificação formal automatizada da consistência da ontologia.

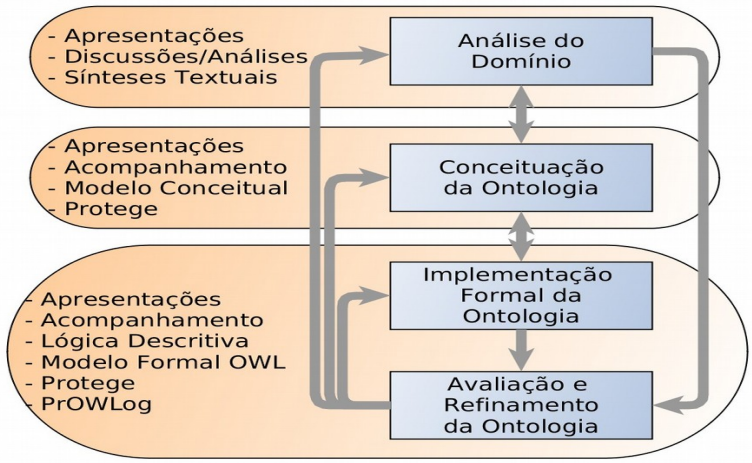

Figura 2. Métodos e ferramentas para ensino de ontologias

3. Ambiente PrOWLog

O PrOWLog é um ambiente lógico-declarativo para programação de aplicações que utilizem ontologias OWL. O ambiente PrOWLog foi desenvolvido visando dois objetivos principais: a) prover um ambiente de ensino para programação de aplicações baseadas em ontologias OWL, mas que leve em conta os aspectos lógicos e declarativos dessas ontologias; b) prover um ambiente de programação declarativo, abstrato e de alto-nível que permita verificar e validar modelos formais especificados em OWL

O PrOWLog oferece um ambiente que permite gerenciar, consultar e alterar ontologias diretamente dentro do interpretador do Prolog. A última versão deste ambiente pode ser baixada de *http://obaa.unisinos.br/prowlog*. O ambiente PrOWLog é implementado como uma extensão do interpretador Prolog, adicionando um conjunto de comandos para executar operações sobre ontologias. Todos os comandos do PrOWLog podem ser incluídos também em programas escritos na linguagem Prolog, provendo assim uma interface de alto nível para acesso e alteração de ontologias OWL dentro do Prolog. O ambiente PrOWLog é totalmente compatível com o *Protege*, assim todas as ontologias geradas no PrOWLog podem ser acessadas e alteradas no Protege e vice-versa, permitindo o uso complementar de ambas ferramentas. PrOWLog também está integrado à motores de inferência (*reasoners*) OWL externos como *Pellet*, *Hermit* ou *Fact++* que podem executar inferências sobre a ontologia armazenada na base semântica, gerando uma ontologia inferida que pode ser consultada.

A gerência básica de ontologias é feita com os comandos *create*, *save, load* e *clear* usados, respectivamente, para criar, salvar em arquivo, lêr de arquivo e apagar (limpar) a ontologia corrente. Os formato de arquivo suportados são: *owl* ou *owldl* ou *prolog*. O formato *owl* é o formato padrão OWL/RDF inteiramente compatível com o a ferramenta *Protégé* e com as bibliotecas JENA e OWLAPI. O formato *owldl* armazena a ontologia na notação compacta similar a sintaxe Manchester para a Lógica Descritiva.

As consultas à ontologia corrente são feitas pelo comando *select*, com uma seguinte muito similar a sintaxe da linguagem SPARQL, ou pelo comando *ask* que é uma versão mais simplificada do comando *select*:

```
select <Var1>,...,<Varn>
  where {<Query1>,...,<Querym>} [from infer] [=> <Results>].
ask {<Query1>,...,<Querym>} [from infer] [=> <Results>].
```

Além das palavras minúsculas para os comandos, a diferença mais importante em relação ao SPARQL é que em PrOWLog as variáveis não são iniciadas por '?' que é a forma de definição de variáveis em SPARQL. No PrOWLog as variáveis <*Var*$_i$> usadas no comando *select* são variáveis Prolog, ou seja, identificadores que começam com letras maiúsculas. A expressão <*Query*$_1$>, ..., <*Query*$_m$> define uma conjunção (um E lógico) de consultas a ontologia corrente. Cada consulta <*Query*$_1$> é uma *consulta de axioma*, formada por uma expressão lógica referente aos axiomas existentes na ontologia. O resultado do comando *select* é apresentado na tela. Porém, se for adicionada a cláusula => <*Result*>, então o resultado da consulta será retornado na variável <*Result*> na forma de uma lista Prolog.

Alterações na ontologia podem ser feitas pelos comandos *insert* e *delete*, que tem uma sintaxe muito similar a SPARQL/Update 1.1:

```
insert {<Axiom1>,...,<Axiomn>}[where {<Query1>,...,<Querym>}].
delete {<Axiom1>,...,<Axiomn>} [where {<Query1>,...,<Querym>}].
```

a lista {<*Axiom*$_1$>,...,<*Axiom*$_n$>} define um novo conjunto de axiomas a ser incluído, no caso do *insert*, ou excluído, no caso do *delete*, da ontologia.

O PrOWLog fornece uma notação compacta, baseada na sintaxe Manchester para a Lógica Descritiva, tanto para a definição quanto para a consulta de axiomas de uma ontologia OWL. Nesta notação, os axiomas de classe podem ser definidos com uma notação infixada. Assim uma classe <*Class*$_1$>pode ser declarada subclasse de outra classe <*Class*$_2$> através de um axioma com o formato: `<Class1> <= <Class2>`. Axiomas de asserção `<Indiv> : <Class>` declaram que o indivíduo <*Indiv*> pertence a classe <*Class*>. A expressão `<Subj> : <Prop>(<Obj>)` define que dois indivíduos <*Subj*> e <*Obj*> estão relacionados pela relação ("*object property*") <*Prop*>. Expressões de classe permitem definir *restrições* adicionais aos axiomas de classe de OWL. A notação compacta de PrOWLog segue a sintaxe Manchester permitindo a intersecção, união e complementação de classes, respectivamente, através das expressões: `<Class1> and <Class2>`, `<Class1> or <Class2>` e `not <Class>`. Também podem ser definidas restrições de quantificação e cardinalidade sobre relações e atributos através de expressões de classe: `<Prop> some <Class>`, `<Prop> only <Class>`, `<Prop> exactly N [of <Class>]`, `<Prop> min N [of <Class>]` e `<Prop> max N [of <Class>]`.

4. Aplicação do Método de Ensino

Um exemplo concreto de aplicação do método de ensino é uma boa forma de ilustrar como este método é utilizado na prática. Neste contexto, considera-se que o aluno está interessado em definir o modelo ontológico para uma aplicação de administração escolar, envolvendo temas como a gestão da alocação das turmas, a distribuição dos professores nas disciplinas e a matricula dos alunos.

4.1. Análise do Domínio

Uma vez que o aluno tenha escolhido o domínio geral, neste caso administração escolar, na etapa de análise de domínio passa-se para a definição dos cenários de motivação para o uso de ontologias neste domínio. São vários os possíveis temas e cenários, porém, para fins de ilustração do método de ensino será considerado apenas um cenário de motivação sobre a definição das turmas, definido textualmente como:

O cenário de definição das turma envolve a reserva dos horários e salas para as turmas, definição dos alunos e professores para cada turma, além da disciplina relacionada a turma.

Para verificação da competência da ontologia neste cenário considera-se que após discussões e análises o aluno tenha elaborado as seguintes questões:

(1) Pode-se identificar e obter informações dos alunos matriculados em uma turma? e o professor da turma?

(2) Há forma de garantir que as turmas tenham um máximo de trinta e um mínimo de um aluno, além de apenas um professor?

(3) Pode-se descobrir em quais turmas um aluno está matriculado e quais turmas um professor é responsável?

(4) Há forma de garantir que não hajam conflitos de salas e horários entre as turmas?

(5) Há como garantir que não hajam conflitos de horários nas turmas de um aluno?

(6) Pode-se obter informações sobre as turmas criadas para uma disciplina?

4.2. Conceituação da Ontologia

Após o estudo e a análise dos vários conceitos envolvidos na definição do cenário e nas questões de competência, o aluno identificou os conceitos de *Turma*, *Aluno*, *Professor*, *Sala*, *Horario* e *Disciplina*, organizados na taxonomia apresentada na Figura 3, gerada com o auxílio do *plugin OWLViz* do Protege.

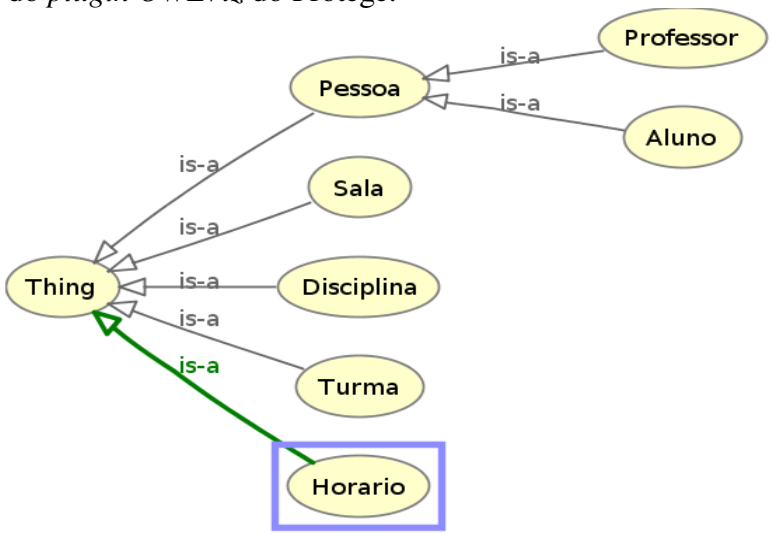

Figura 3. Taxonomia de conceitos do cenário de definição de turmas

As relações ("*object properties*") e atributos ("*data properties*") identificados como necessários e suficientes para responder todas as questões de competência são apresentadas na Figura 4, tal como extraídas das telas do *Protege*.

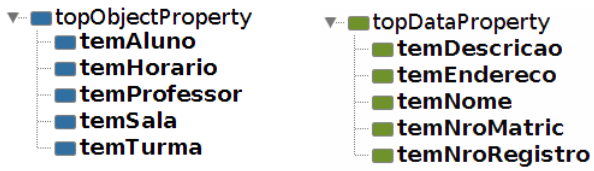

Figura 4. Relações e atributos para o cenário de definição de turmas

O aluno considera que uma turma é composta pelo professor responsável, por seus alunos, pelo horário e sala alocados para a turma. Isso implica que as relações *temProfessor*, *temAluno*, *temSala* e *temHorario* são relações de composição deste cenário, ou seja, formam parte da mereologia da ontologia. Os atributos *temNome* e *temEndereco* permitem obter as informações básicas sobre professores e alunos, enquanto, que *temNroMatric* e *temNroRegistro*, são específicos, respectivamente, para alunos e para os professores. A descrição de uma disciplina pode ser obtida pelo atributo *temDescrição*.

4.3. Implementação Formal da Ontologia

A ontologia definida nas Figuras 3 e 4 com o uso da ferramenta Protege pode ser salva em formato RDF/XML e então lida pelo PrOWLog. A Figura 5 mostra os axiomas dessa ontologia na notação compacta de OWL utilizada pelo PrOWLog.

```
class('Aluno').  class('Disciplina'). class('Horario'). class('Sala').
class('Pessoa'). class('Professor'). class('Turma'). 'Aluno'<='Pessoa'.
'Professor'<='Pessoa'.  rel(temAluno). rel(temHorario). rel(temSala).
rel(temTurma). rel(temProfessor). attr(temNome). attr(temDescricao).
attr(temNroMatric). attr(temEndereco). attr(temNroRegistro).
```

Figura 5. Formalização inicial da ontologia na notação compacta de OWL

Note que além das declarações de quais são as classes, relações e atributos da ontologia, a formalização apresentada na Figura 5 apenas inclui axiomas que definem a especialização da classe Pessoa em Aluno ($\mathsf{Aluno} \sqsubseteq \mathsf{Pessoa}$) e Professor ($\mathsf{Professor} \sqsubseteq \mathsf{Pessoa}$). Assim muitos outros axiomas devem ser incluídos para que as questões de competência possam ser respondidas de forma apropriada. Após a introdução aos conceitos de Lógica Descritiva o aluno considera que as relações de composição da classe *Turma* podem ser formalizadas pelos seguintes axiomas:

$\mathsf{Turma} \sqsubseteq (\exists \mathsf{temProfessor}.\mathsf{Professor} \sqcap \exists \mathsf{temAluno}.\mathsf{Aluno})$
$\mathsf{Turma} \sqsubseteq (\exists \mathsf{temSala}.\mathsf{Sala} \sqcap \exists \mathsf{temHorario}.\mathsf{Horario})$

Tais axiomas podem ser representados em PrOWLog pelas expressões:

```
'Turma' <= (temProfessor some 'Professor' or temAluno some 'Aluno')
'Turma' <= (temSala some 'Sala' or temHorario some 'Horario')
```

Estes axiomas podem, então, ser incluídos na ontologia simplesmente através do seguinte comando PrOWLog:

```
add{'Turma'<=(temProfessor some 'Professor' or temAluno some 'Aluno'),
    'Turma' <= (temSala some 'Sala' or temHorario some 'Horario') }.
```

Axiomas específicos que definem os tipos de dados dos atributos como do tipo *xsd:string* também foram incluídos pelo aluno:

```
add {   'Pessoa'<=temNome some xStr, 'Pessoa'<=temEndereco some xStr,
  'Aluno'<=temNroMatric some xStr,
  'Professor'<=temNroRegistro some xStr }.
```

4.4. Avaliação e Refinamento da Ontologia

Após a introdução dos axiomas específicos para as relações de composição e para os tipos dos atributos, o aluno considera que a ontologia está apta para o processo de avaliação e refinamento. Para fins de verificação das consultas são introduzidos os axiomas da TBox, incluindo asserções específicas para os indivíduos, além de asserções sobre relações e atributos. A Figura 6 mostra um pequeno subconjunto destes axiomas.

```
indv(al001).  indv(al002). indv(pr01).  indv(pr02).
indv(s10).    indv(seg_m). indv(ter_m). indv(t101).
al001:'Aluno'.    al002:'Aluno'. pr01:'Professor'.   pr02:'Professor'.
t101:'Turma'.     s10:'Sala'.      seg_m:'Horario'.  ter_m:'Horario'.
al001:temNome('Aluno 1').         al001:temNroMatr('10001').
al002:temNome('Aluno 2').         al002:temNroMatr('10002').
pr01:temNome('Professor A').      pr01:temNroRegistro('101').
t101:temProfessor(pr01).
t101:temAluno(al001).      t101:temAluno(al002).
t101:temSala(s10).         t101:temHorario(seg_m).
```

Figura 6. Asserções (TBox) da ontologia na notação compacta de OWL

A consistência da ontologia pode ser verificada através da ativação de um motor de inferência (*reasoner*). O aluno utilizou o *reasoner* Pellet e fez essa verificação através dos comandos:

```
set reasoner to pellet.    start reasoner.    ask consist from infer.
```

obtendo *true* como resposta, o que indica que a ontologia, pelo menos até agora é consistente.

Seguindo as questões definidas na Seção 4.1, o aluno considera que a questão (1) sobre a identificação e obtenção de informações sobre os alunos e professor da turma podem ser facilmente respondidas por consultas PrOWLog similares a seguinte, que busca essas informações para a turma *t101*:

```
select N,M where {t101:temAluno(A), A:temNome(N), A:temNroMatr(M)}.
select N,R where{t101:temProfessor(P),P:temNome(N),P:temNroRegistro(R)}.
```

A formalização da questão (2) mostrou algumas dificuldades. Na formalização atual da ontologia a verificação do número máximo de alunos tem que ser feito por meios extra-lógicos: um programa pode ser construído para contar os resultados obtidos com uma consulta sobre os alunos matriculados em uma dada turma. Depois de conversar com o professor o aluno descobre que existe uma maneira melhor de se garantir isso em OWL, através da adição de restrições de cardinalidade para *temAluno* e para *temProfessor*. Para tanto o aluno executa os seguintes comandos:

```
add {   'Turma'<=temAluno max 30 of 'Aluno',
  'Turma'<=temProfessor exact 1 of 'Professor'}.
```

para verificar essas restrições também é adicionada a seguinte asserção:

```
add {t101:temProfessor(pr02)}.
```

logo após, o *reasoner* pode ser ressincronizado por um comando `resync reasoner` e a consistência verificada novamente.

Entretanto, para surpresa do aluno o *reasoner* informa que a ontologia ainda está consistente. Após revisar várias vezes os axiomas, o aluno explica a questão ao professor. Depois de uma análise conjunta, o problema é identificado: diferente do Prolog, OWL trabalha com a pressuposição de mundo aberto (OWA – *Open World Assumption*). Pela OWA não é possível inferir que *pr01* e *pr02* designam necessariamente dois professores (indivíduos) diferentes. Porém, se isso for explicitamente afirmado na ontologia, então a ontologia deve se tornar inconsistente. De fato, após a inclusão dessa afirmação através do comando `add {pr01 \= pr02}` e de uma nova ressincronização do *reasoner*, a verificação de consistência retorna falso.

As demais questões de competência são tratadas de forma similar até que a competência da ontologia seja verificada em relação ao cenário de definição de turmas no domínio de administração escolar.

5. Considerações Finais

O método de ensino para projeto e desenvolvimento de ontologias apresentado neste trabalho vem sendo utilizado há mais de dois anos. Durante este período os alunos conceberam e verificaram ontologias através desse método para os vários domínios de conhecimento, incluindo, por exemplo, ontologias para o controle de versões de software, administração de recursos humanos, representação de relacionamentos pessoais, análise de redes sociais e gerenciamento de carreiras. O método está em estado de constante aperfeiçoamento. Um dos pontos mais importantes sendo estudados atualmente é a transformação da ontologia de uma estrutura formal consistente capaz de representar os conhecimentos de um dado domínio, para a base de uma nova aplicação de software. Existem inúmeros problemas nessa passagem que merecem uma grande atenção. O suporte a programação OWL é oferecido por *frameworks* ou APIs de

programação como OWL-API, JENA API e Protege API. Porém, os mecanismos de programação suportados nestas APIs seguem uma abordagem estrutural e algorítmica convencional. Eles ignoram quase completamente os aspectos declarativos e lógicos de OWL, o que torna o acesso e manipulação dos elementos da ontologia uma tarefa desnecessariamente tediosa e complexa. A linguagem de consulta SPARQL oferece uma alternativa possível, principalmente para aplicações voltadas para bancos de dados. Porém o suporte de SPARQL em linguagens de programação segue a mesma abordagem de uso de SQL nessas linguagens, ou seja, a programação de consultas SPARQL em uma linguagem como Java não é integrado a sintaxe da linguagem de programação nem seu uso é transparente ao programador.

Referências

Baader, F. ;McGuinness, D.L; Nardi. D.; Patel-Schneider, P.F. (Eds.) (2003) *The Description Logic Handbook: Theory, Implementation, and Applications*. New York.

Balog-Crisan, R.; Roxin, I. (2008) Semantic Learning Content Management System. *Procs. of IADIS Int. Conf. on e-Learning*.

BioPortal. (2016) *Welcome to BioPortal, the world's most comprehensive repository of biomedical ontologies*. Disponível em <*http://bioportal.bioontology.org/ontologies*>

Bittencourt, I. I.; Costa, E.; Silva, M.; Soares, E. (2009) A computational model for developing semantic web-based educational systems. *Knowledge-Based Systems*, v. 22, n.4, p.302–315.

Carrion, J.S.; Gordo, E.G.; Sanchez-Alonso, S. (2007) Semantic learning object repositories. *Int. J. of Continuing Eng. Education and Life Long Learning*, v. 17, i. 6.

Dicheva, D.; Mizoguchi, R.; Greer. J. (Eds.) (2009) *Semantic Web Technologies for e-Learning*. IOS Press.

Dzbor, M.; Drajpathak, D. (2009) Comparative Evaluation of ASPL, Semantic Platform for e-Learning. In: Dicheva, D. et al. (Eds.) *Semantic Web Technologies for e-Learning*. IOS Press.

Gluz, J. C.; Vicari, R. M. (2010) MILOS: Infraestrutura de Agentes para Suporte a Objetos de Aprendizagem OBAA. *Anais do XXI SBIE* (SBIE 2010).

Gómez-Pérez, A.; Fernández-López, M.; Corcho, O. (2004) *Ontological Engineering*. Springer.

Grimm, S.; Abecker, A. Volker, J.; Studer, R. (2011) Ontologies and the Semantic Web. In: Domingue, J. et al. (Eds.) *Handbook of Semantic Web Technologies*. Springer.

Hayashi, Y.; Isotani, S.; Bourdeau, J.; Mizoguchi, R. (2009) Toward a Learning/Instruction Process Model for Facilitating the Instructional Design Cycle. *Procs. of WCCE 2009*.

Isotani, S.; Mizoguchi, R.; Bittencourt, I.; Costa, E. (2008) Web 3.0 - Os Rumos da Web Semântica e da Web 2.0 nos Ambientes Educacionais. *Anais do XIX SBIE*, Fortaleza.

Kholief, M.; Nada, N.; Khedr, W. (2012) Ontology-Oriented Inference-Based Learning Content Management System. *Int. J. of Web & Semantic Technology* (IJWesT) v.3, n.3, July.

Melis, E.; Goguadze, G.; Libbrecht, P.; Ullrich. C. (2009) ActiveMath – a Learning Platform With Semantic Web Features. In: Dicheva, D. et al. (Eds.) *Semantic Web Technologies for e-Learning*. IOS Press.

Porto, F.; Moura, A.M.C.; da Silva, F.J.C; Fernandez, A.P. (2007) The ROSA project: leveraging e-learning to a semantic layer. *Int. J. Knowledge and Learning*, v. 3, n. 1.

Pahl, C.; Javed, M.; Abgaz, Y.M. (2013) Ontology Evolution for Learning Content Management Systems. *Procs of World Conf. on Educational Media and Technology* (EdMedia 2013)

Silva, M.; Barros, H.; Veras, D.; Pacca, H.; Ibert, I.; Barros, E.; Silva, A. (2009) Modelando um Sistema Educacional de MMC sob a perspectiva da Web Semântica. *Anais SBIE 2009*.

Simperl, E.; Bontas, P.; Christoph, T. (2006) Ontology Engineering: A Reality Check In: *Procs. of OTM Confederated Int. Conf., CoopIS, DOA, GADA, and ODBASE.* Montpellier, France.

Sure, Y.; Tempich, C.; Vrandecic, D. (2006) Ontology Engineering Methodologies. In: Davies, J.; Studer, R.; Warren, P. (Eds.) *Semantic Web Technologies Trends and Research in Ontology-based Systems*. John Wiley & Sons.

W3C. (2011). *W3C Rec.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax* (2nd Ed). W3C.

W3C. (2009) *W3C Rec.: OWL 2 Web Ontology Language: Manchester Syntax*. W3C.