Aplicação da Reengenharia de Software na Construção Acelerada de Ontologias

Regina C. Cantele [1], **Diana F. Adamatti** [2]*;
Maria A. G. V. Ferreira [1], **Jaime S. Sichman** [2] †

[1]InterLab - Laboratório de Tecnologias Interativas
Escola Politécnica da Universidade de São Paulo
[2]LTI - Laboratório de Técnicas Inteligentes
Escola Politécnica da Universidade de São Paulo
{regina.cantele,diana.adamatti}@poli.usp.br
{maria.alice.ferreira,jaime.sichman}@poli.usp.br

Abstract. *Ontologies are a basic building block of the Semantic Web. As a consequence, a great number of researchers are working in method and techniques to build ontologies through automatic or semi-automatic processes, which perform knowledge acquisition from texts, dictionaries and structured and semi-structured knowledge bases. On the other hand, reverse engineering, when applied to software engineering, uses a collection of theories, methodologies and techniques to support information abstraction and extraction from a piece of software. This paper presents the results of an initial study, which uses reverse engineering techniques applied to ontology engineering, whose goal is to reduce the time consuming task of ontology creation. A case study was developed to test this proposal, which is applied in a educational software, using some OMG standards (UML, MOF and XMI). The ontology obtained represents old knowledge about a specific domain and it can help engineers to build a final ontology.*

Resumo. *Ontologias são blocos básicos na construção da Web Semântica. Consequentemente, um grande número de pesquisadores estão trabalhando em métodos e técnicas para construir ontologias através de processos automáticos ou semi-automáticos, que realizam aquisição de conhecimento em textos, dicionários e bases de conhecimento estruturadas ou semi-estruturadas. Por outro lado, a engenharia reversa, quando aplicada à engenharia de software, utiliza uma coleção de teorias, metodologias e técnicas para suportar e extrair abstrações das informações de um fragmento de software. Este artigo apresenta os resultados de um estudo inicial, em que se usam técnicas de engenharia reversa aplicadas à engenharia de ontologias, cujo objetivo é reduzir o tempo de desenvolvimento de uma ontologia. Um exemplo foi desenvolvido para testar esta proposta em um software educacional, utilizando alguns padrões da OMG (UML, MOF e XMI). A ontologia obtida representa o conhecimento "antigo" sobre um domínio específico e pode ajudar os engenheiros a construir uma ontologia final.*

1. Introdução

Andrônico de Rodes, por volta de 50 a.C., recolheu e classificou as obras de Aristóteles que, durante muitos séculos, haviam ficado dispersas e perdidas. Isto mostra que, desde este período, o homem deseja organizar o conhecimento coletado: pelas organizações, comunidades científicas e indivíduos em diferentes fontes de informação - atualmente, na Web ou em banco de dados legados -, e transformá-lo em informações úteis, através do uso de ontologias.

De acordo com Guarino [11] e Gruber [10], uma ontologia representa um vocabulário comum de um domínio. Assim, define o significado dos termos e as relações entre eles, organizando-os em uma taxonomia. Contém primitivas de modelagem como classes, relações, funções, axiomas e instâncias. Existem linguagens tradicionais para sua representação como: CYCL [17], Ontolingua [7], F-Logic [15] ou CML [24], e linguagens padrões para Web como: OIL (*Ontology Inference Layer*) [1], DAML+OIL (*DARPA Agent Markup Language*) [2], RDF(s) (*Resource Description Framework*), XOL (*Ontology Exchange Language*)

*Financiado pelo CNPq.
†Parcialmente financiado pelo CNPq, processo no. 301041/95-4.
[1]http://www.ontoknowledge.org/oil/
[2]http://www.daml.org

[14], SHOE (*Simple HTML Ontology Extensions*) [18], XTM (*XML Topic Maps*) e OWL (*Ontology Web Language*)[3]. É importante ressaltar que existem diferentes conexões entre os componentes da ontologia, seus paradigmas de representação do conhecimento e suas linguagens de representação.

O estudo de ontologias está fortemente embasado em princípios da Inteligência Artificial. Esta, por sua vez, está emprestando sua fundamentação a muitas outras disciplinas, entre as quais a Engenharia de Software. Desde seu aparecimento, na década de 70, a Engenharia de Software vê-se imersa em uma crise que já se tornou crônica, caracterizada pela necessidade de produzir novos sistemas de informação, mais poderosos e complexos, e manter sistemas antigos, sem interromper a operação normal. Assim, os engenheiros de software precisam apoiar-se em ferramentas especializadas para garantir produtividade e qualidade, condizentes com as exigências do mercado [23]. A engenharia reversa, um de seus ramos, utiliza uma coleção de teorias, metodologias e técnicas para extrair e abstrair informações de um software existente, que necessite ser reconstruído, produzindo documentos consistentes, quer seja a partir do código fonte, ou através da adição de conhecimento e experiência, que não poderiam ser automaticamente reconstruídos a partir deste código. Os sistemas existentes possuem, muitas vezes, somente o código executável. Tais sistemas não podem ter sua operação interrompida porque a informação que gerenciam ou encapsulam é fundamental para os negócios da empresa.

Os engenheiros de ontologias utilizam a metodologia inerente ao projeto do qual fazem parte, tais como *Enterprise Ontology* [25], *TOVE* (*Toronto Virtual Enterprise*) [10] e *Methontology* [8]. Estas metodologias apresentam ciclos de vida distintos, mas as fases de aquisição e formalização das ontologias sempre aparecem em todos os ciclos. Desta forma, na aquisição constrói-se um modelo conceitual e na formalização um modelo formal. O objetivo deste artigo é relatar uma experiência em que se combina a reengenharia de sistemas com a engenharia de ontologias para reduzir o tempo dispendido na construção de ontologias. Desta forma, a seção 2 apresenta uma pequena revisão sobre ontologias, a seção 3 uma revisão da reengenharia e a seção 4 a proposta para combinação de engenharia reversa e ontologias. Na seção 5 são descritos exemplos de aplicação desta proposta e na seção 6 estão as conclusões do artigo.

2. Ontologias

Pesquisas em ontologias têm origem na filosofia com a natureza e organização das "coisas". Uma ontologia, segundo Gruber [10], é uma especificação explícita dos objetos, conceitos e outras entidades que se assume existirem em uma área de interesse, além das relações entre estes conceitos e restrições, expressos através de axiomas. Em Ciência da Computação, o termo ontologia refere-se a um artefato de engenharia, constituído por um vocabulário específico que descreve um modelo particular do mundo, adicionando um conjunto explícito de suposições, relacionando os significados das palavras no vocabulário, usualmente, organizados em taxonomias [19]. A ontologia deve fazer uma especificação formal de uma área de conhecimento. Cinco componentes foram definidos para esta formalização [10]:

- 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; a cardinalidade é sempre n:n;
- Funções: são um caso especial de relações; a cardinalidade é n:1;
- Axiomas: são sentenças que são sempre verdadeiras;
- Instâncias: são utilizadas para representar os elementos do domínio.

O aparecimento da Web Semântica marcou outro estágio no campo das ontologias. De acordo com Berners-Lee [3], a "Web Semântica não é uma Web separada, mas uma extensão da atual, na qual a informação é utilizada com significado bem definido e não-ambíguo, integrada, aumentando a capacidade dos computadores para trabalharem em cooperação com as pessoas". A Web Semântica é coordenada pelo consórcio W3C (*World Wide Web Consortium*) e apoiada pela indústria de software. Este consórcio propôs uma arquitetura com sete camadas - Unicode e URI, XML + NS + XML Schema, RDF + RDF Schema, Ontology vocabulary, Logic, Proof e Trust - para construir aplicações que envolvam a Web Semântica. Esta arquitetura define as tecnologias necessárias para a representação formal de ontologias, entre elas, a linguagem OWL (*Ontology Web Language*). Alguns pesquisadores [6, 2] uniram o formalismo existente para a representação de ontologias com a notação UML (*Unified Modeling Language*). A UML é mantida pelo OMG (*Object Management Group*), um grupo que fornece diretrizes para a indústria de software através de especificações de padrões, cuja missão é promover a teoria e a prática da tecnologia de objetos para o

[3] http://www.w3.org/2004/OWL

desenvolvimento de sistemas distribuídos. A UML é uma linguagem gráfica, utilizada em Engenharia de Software para modelar sistemas orientados a objetos. Além de definir a notação gráfica (conjunto de símbolos padrão), especifica a semântica destes símbolos de tal forma que o modelo conceitual para o sistema possa ser compreendido por todos, facilitando, assim, a comunicação efetiva entre as pessoas envolvidas na sua utilização [22]. Além disso, a UML é extensível; é essa característica que tem permitido a sua aplicação em ontologias [4].

3. Reengenharia: engenharia reversa e engenharia progressiva

Para Chikofsky e Cross [4], a reengenharia, conhecida também como renovação ou reconstrução, é o exame de um sistema de software, a fim de reconstituí-lo em uma nova forma, e a subseqüente implementação dessa nova forma. Um processo de reengenharia geralmente inclui alguma forma de engenharia reversa, seguida por uma forma de engenharia progressiva ou reestruturação [13].

Define-se engenharia reversa como uma coleção de teorias, metodologias e técnicas capazes de suportar a extração e abstração de informações de um software existente, produzindo documentos consistentes, quer seja a partir do código fonte, ou através da adição de conhecimento e experiência que não podem ser, automaticamente, reconstruídos a partir do código [4]. A engenharia reversa de um sistema que deve ser reconstruído pode ser realizada de diversas maneiras: através dos códigos-fontes ou executáveis, através do repositório de dados da ferramenta CASE utilizada no seu desenvolvimento ou através dos dicionários dos bancos de dados utilizados pelo sistema. Atualmente, a representação obtida pelo processo de engenharia reversa segue padrões de mercado, tais como UML, MOF e XMI, de forma a ser compreendida facilmente pelo maior número possível de pessoas e ferramentas. O padrão OMG-MOF (*Meta Object Facility*)[20] é um conjunto de serviços projetados para suportar a administração de metadados. Metadado é um dado que descreve outro dado, com um nível maior de abstração. Exemplos de metadados são os tipos de dados nas linguagens de programação, de definições de interfaces dos componentes no paradigma de Orientação a Objetos, diagramas de análise e projeto e esquemas SQL, que descrevem a estrutura de bancos de dados relacionais. Já o padrão OMG-XMI (*XML Metadata Interchange*) representa um mecanismo padrão para troca de dados entre as várias ferramentas, repositórios e metadados. A padronização garante a troca de metadados entre ferramentas de modelagem baseadas no padrão OMG-UML e os repositórios de metadados baseados em OMG-MOF, em ambientes heterogêneos distribuídos. O XMI tem sido usado, também, para interpretar artefatos UML, artefatos de bases de dados e *data warehouse*, definições de interfaces CORBA (*Common Object Request Broker Architecture*)[5] e de interfaces e classes JAVA [21].

Define-se engenharia progressiva como sendo o processo tradicional de produzir software a partir de abstrações de alto nível, através de projetos independentes de implementação, gerando a versão física do sistema [4]. Na engenharia progressiva para desenvolvimento de ontologias, alguns projetos implementaram metodologias próprias, tais como o *Enterprise Ontology* [25], TOVE [9] e *Methontology* [5]. Uma metodologia se caracteriza pela adoção de um processo para o desenvolvimento do sistema, apoiado por ferramentas que auxiliam as várias etapas, nas quais se liberam artefatos, conforme o planejamento do projeto. O desenvolvimento e o uso de metodologias para a construção de ontologias é fundamental, na medida que retiram o caráter subjetivo desta atividade. A área de Engenharia Ontológica estuda os aspectos relacionados a tal construção, bem como o desenvolvimento de sistemas que utilizem ontologias em sua estrutura [19].

4. Ontologia e Reengenharia

O desenvolvimento de software, a partir da década de 90, passou a ter como foco a qualidade. Sob esse prisma, a adoção de uma metodologia é fundamental [23]. Este artigo propõe as etapas descritas a seguir para a obtenção de uma ontologia, para um sistema que venha a evoluir na direção da Web Semântica, tomando-se como base as informações disponibilizadas em sistemas de software existentes. As seguintes etapas são propostas:

1. Aplicar Engenharia Reversa ao sistema existente para a obtenção de metadado da ontologia: a aplicação de técnicas de reengenharia permite obter-se uma versão inicial da ontologia;
2. Representar a ontologia obtida em UML estendida;
3. Adotar a Engenharia Progressiva: aplicar uma das metodologias existentes para o desenvolvimento de uma ontologia completa.

[4] http://www.omg.org/ontology
[5] http://www.omg.org/corba

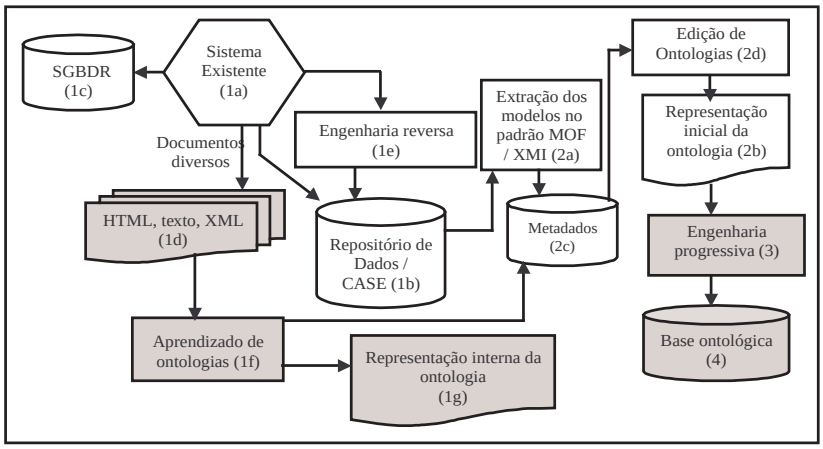

Figura 1: Combinação da reengenharia de sistemas e engenharia de ontologias

A seguir, exemplifica-se a proposta, supondo-se como ponto de partida para a construção da ontologia um sistema existente. Conforme apresentado na Figura 1, o conhecimento do sistema existente (1a) está geralmente presente nos metadados das ferramentas utilizadas na sua concepção e implementação, que podem ser, por exemplo, o dicionário de dados do gerenciador de banco de dados relacional utilizado (1c), o repositório da ferramenta CASE utilizada na sua construção (1b) ou os documentos referentes ao sistema (help, planilhas, textos, etc.) (1d). Quando o sistema existente (1a) não utilizou uma ferramenta CASE na sua construção, o repositório de dados (1b) não existe, e então é necessário realizar a Engenharia Reversa (1e) sobre o dicionário de dados do gerenciador de banco de dados (SGBDR) (1c) para obter-se o diagrama de classes da UML com suas classes, atributos, associações e cardinalidade, atualizando (1b). Outros diagramas do sistema existente podem ser obtidos através de (1e), tais como Diagramas de Seqüência, Diagramas de Estados e Diagrama de Casos de Uso. Os documentos diversos do sistema existente (1d), também podem produzir uma representação interna da ontologia (1g) aplicando uma das técnicas de aprendizado de ontologias (1f).

O processo deve ser realizado por ferramentas implementadas com o padrão MOF (2a) para que o resultado final obtido seja apresentado no padrão MOF-XMI. É a transformação de uma forma de representação em outra, de mesmo nível de abstração relativo, preservando o comportamento externo do sistema (funcionalidade semântica). Assim, poderá ser editado a primeira representação conceitual da ontologia (2d), de forma compreensível pelos engenheiros de ontologias, sendo o primeiro modelo da ontologia no formato UML (2b), ou seja, uma primeira coleção de conceitos e relações sobre o domínio a ser detalhado. Os resultados obtidos nos processos (1f) e (2a) geram os metadados da ontologia (2c).

Em um segundo momento, aplica-se a Engenharia Progressiva (3), a partir da primeira versão proposta (2b); os engenheiros de ontologias seguem uma da metodologias citadas na seção 3, garantindo a utilização dos padrões do W3C descritos na seção 2. A utilização de metodologias garante que a base ontológica (4) será consistente e coerente com o domínio a ser representado e com o modelo atual do sistema.

5. Exemplos de Utilização da Proposta

Os exemplos em que se aplicou a proposta foram dois sistemas educacionais de ensino a distância utilizados por grandes universidades brasileiras: o Teleduc[6], da Universidade Estadual de Campinas e o CoL[7], utilizado pela Escola Politécnica da Universidade de São Paulo.

Para realizar a primeira etapa da proposta apresentada na seção 4 (Engenharia Reversa), utilizou-se os metadados das bases de dados relacionais dos sistemas (1c), sendo SQL Server no caso do CoL e MySQL no caso do Teleduc. Com a ferramenta MagicDraw[8], que implementa os padrões UML-MOF-XMI, realizou-se a engenharia reversa (1e), obtendo-se o repositório de dados (1b). Realizou-se então a segunda etapa (Representar a ontologia obtida em UML estendida), a partir da extração dos modelos no padrão MOF-XMI (2a), obtendo-se os metadados (2c). Em seguida, realizou-se o processo de edição de ontologias (2d) com a ferramenta de desenvolvimento de ontologias Protégé[9], que importa o padrão XMI obtido (2a), editando a representação inicial da ontologia (2b), em UML estendida.

[6] http://teleduc.nied.unicamp.br/teleduc
[7] http://col.larc.usp.br
[8] http://www.magicdraw.com
[9] http://protege.stanford.edu

Para a realização da terceira etapa (Aplicar Engenharia Progressiva) escolheu-se a *Methontology* [5], pois indica de forma detalhada as atividades a serem realizadas para a construção de ontologias, bem como indica a seqüência/ordem e o nível de detalhamento em que tais atividades devem ser executadas. Resumidadamente, esta metodologia está dividida em três grandes grupos de atividades:

1. Atividades de Gerenciamento de Projeto: incluindo planejamento, controle e garantia de qualidade;
2. Atividades de Desenvolvimento Orientado: inclui especificação, conceitualização, formalização e implementação. A especificação define o porquê da ontologia estar sendo construída (utilização e usuários finais). A conceitualização estrutura o domínio de conhecimento em um modelo conceitual. A formalização transforma o modelo conceitual em um modelo formal ou semi-computável. A implementação transforma modelos computáveis em linguagens computacionais;
3. Atividades de Suporte: inclui aquisição de conhecimento, avaliação, integração, documentação e gerenciamento de configurações.

O resultado das etapas 1 e 2 da proposta - uma representação inicial da ontologia - visa ajudar/facilitar o trabalho de conceitualização e formalização, descritos na atividade 2 da *Methontology* , referentes à etapa 3 da proposta (Engenharia Progressiva) que não foi foco de estudo neste artigo bem como os processos em cinza na Figura 1.

5.1. Execução dos Processos

Para um melhor entendimento da execução dos processos e do poder da utilização dos padrões, com as ferramentas descritas, foram selecionadas três tabelas, de um total de 600, do sistema educacional CoL: "Disciplinas", "Modulo"e "Composicao". Na Figura 2, estão os comandos SQL ANSI de criação destas três tabelas. Através deste *script*, foi realizada a reengenharia com a ferramenta MagicDraw, obtendo o diagrama classes UML, conforme Figura 3. A partir do diagrama de classes UML, sem nenhuma alteração, foi realizada a exportação do modelo em um arquivo no padrão XMI do qual a (Figura 4) apresenta alguns fragmentos. O arquivo XMI foi importado na ferramenta Protégé utilizando a linguagem OWL, resultando uma representação inicial da ontologia, apresentada na Figura 5 (visualização no plug-in gráfico do Protégé *Jambalaya*).

```
CREATE TABLE    Disciplinas  (
          id    int  NOT NULL ,
          codigo    char  (8) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
          nome    varchar  (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
          descricao    text  COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
          professor    int  NULL ,
          link_glossario    varchar  (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
          link_bibliografia    varchar  (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
          link_sumario    varchar  (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
          link_extra_nome    varchar  (40) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
          link_extra    varchar  (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
          grupo_padrao    int  NULL ,
          versao    char  (40) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON   PRIMARY  TEXTIMAGE_ON   PRIMARY;

CREATE TABLE    Modulos  (
          id    int  NOT NULL ,
          nome    varchar  (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
          descricao    text  COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
          url    varchar  (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
          professor    int  NULL ,
          teste    bit  NULL ,
          subdiretorio    char  (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON   PRIMARY  TEXTIMAGE_ON   PRIMARY;

CREATE TABLE    Composicao  (
          disciplina    int  NOT NULL ,
          modulo    int  NOT NULL ,
          ordem    int  NULL ,
          separador    char  (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
          texto    text  COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON   PRIMARY  TEXTIMAGE_ON   PRIMARY;

ALTER TABLE Composicao ADD
  CONSTRAINT FK_Composicao_Disciplinas FOREIGN KEY (disciplina) REFERENCES Disciplinas (id),
  CONSTRAINT FK_Composicao_Modulos FOREIGN KEY(modulo) REFERENCES Modulos (id);
```

Figura 2: Subconjunto do *script* de criação SQL, obtido da base de dados CoL

5.2. Análise dos Resultados

A Tabela 1 apresenta as transformações que ocorreram desde o banco de dados até a representação inicial da ontologia, sem perda de informação.

Apesar da semântica de cada um dos modelos - relacional, UML e ontologia - ser diferente, o conhecimento representado não é perdido nas transformações realizadas. Por exemplo, uma "Disciplina" tem um código, um nome, um professor e uma versão; um "Modulo" tem um nome e um professor; já uma "Composicao", é uma associação entre "Disciplinas"e "Modulos". O exemplo tratou de poucas tabelas do banco de dados, parecendo à primeira vista que uma reengenharia manual seria mais fácil de ser realizada. Porém, ao considerar o número total de tabelas (600), a reengenharia manual não será trivial, mostrando que a utilização do processo descrito é mais eficiente. O uso de mnemônicos significativos dos sistemas a

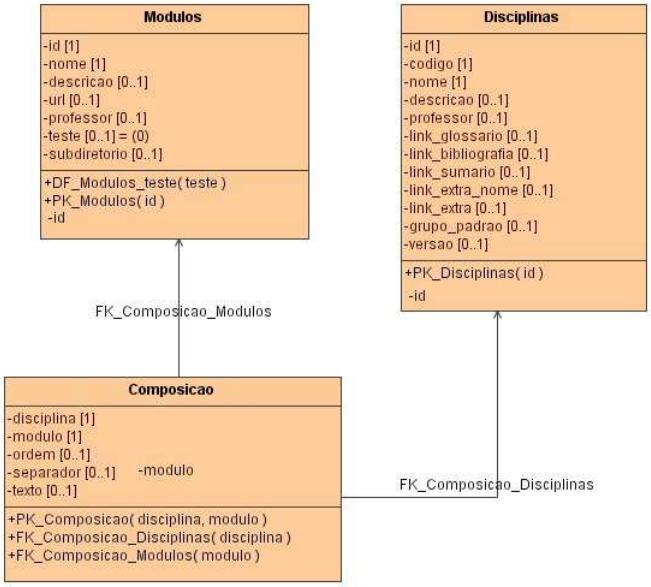

Figura 3: Subconjunto do diagrama UML, obtido a partir do *script* de criação SQL

```
<UML:Class xmi.id='_a6e02c6_1101163156012_741874_58' name='Modulos' visibility='public' isRoot='fals
  <UML:ModelElement.stereotype>
   <UML:Stereotype href='DDL_Profile.xml|magicdraw_1048146927535_885898_81'>
    <XMI.extension xmi.extender='MagicDraw UML 8.0' xmi.extenderID='MagicDraw UML 8.0'>
     <referentPath xmi.value='DDL Profile::table'/>
      </XMI.extension>
      </UML:Stereotype>
        </UML:ModelElement.stereotype>
          <UML:Classifier.feature>
             <UML:Attribute xmi.id='_a6e02c6_1101163156022_74202_59' name='id' visibility='private'
            <UML:StructuralFeature.multiplicity>
               ..................

<UML:Class xmi.id='_a6e02c6_1101163155391_344574_11' name='Disciplinas' visibility='public' isRoot='
  <UML:ModelElement.stereotype>
   <UML:Stereotype href='DDL_Profile.xml|magicdraw_1048146927535_885898_81'>
     <XMI.extension xmi.extender='MagicDraw UML 8.0' xmi.extenderID='MagicDraw UML 8.0'>
      <referentPath xmi.value='DDL Profile::table'/>
        </XMI.extension>
          </UML:Stereotype>
            </UML:ModelElement.stereotype>
            <UML:Classifier.feature>
              <UML:Attribute xmi.id='_a6e02c6_1101163155452_279782_12' name='id' visibility='private
                <UML:StructuralFeature.multiplicity>
                 ..................


<UML:Class xmi.id='_a6e02c6_1101163156022_969036_66' name='Composicao' visibility='public'
 <UML:ModelElement.stereotype>
  <UML:Stereotype href='DDL_Profile.xml|magicdraw_1048146927535_885898_81'>
   <XMI.extension xmi.extender='MagicDraw UML 8.0' xmi.extenderID='MagicDraw UML 8.0'>
     <referentPath xmi.value='DDL Profile::table'/>
      </XMI.extension>
        </UML:Stereotype>
          </UML:ModelElement.stereotype>
            <UML:Classifier.feature>
               <UML:Attribute xmi.id='_a6e02c6_1101163156022_916897_67' name='disciplina'
                  ..................
```

Figura 4: Subconjunto do padrão XMI, obtido do diagrama UML

serem reestruturados auxilia muito no conhecimento representado. Isso pode ser observado pelos nomes das tabelas (por exemplo: "Disciplinas") ou pelos nomes de colunas (por exemplo: nome, código, versão, etc.). A mesma seqüência de execução dos processos foi realizada para o caso do Teleduc (com aproximadamente 20 tabelas), no qual foram obtidas as mesmas conclusões, em relação às informações geradas.

6. Conclusões e Trabalhos Futuros

Este artigo apresentou a viabilidade conceitual e prática de integração dos diversos padrões e técnicas associados aos temas de reengenharia de sistemas e engenharia de ontologias. Tanto os padrões da OMG (UML, MOF, XMI), quanto os padrões da W3C (XML e OWL), são reconhecidos, aceitos e utilizados pela comunidade de Tecnologia da Informação. A engenharia de ontologias pode se valer disso para obter representações iniciais de seus artefatos de forma a aproveitar o conhecimento já representado. Deve-se considerar que a semântica de um termo pode variar de um contexto para outro, de um lugar para outro e mesmo de uma pessoa para outra. Devido a essa heterogeneidade semântica, a engenharia de ontologias não é uma atividade trivial e requer tempo, disponibilidade e consenso dos especialistas. Logo, a idéia do aproveitamento do que já foi representado no passado, através da reengenharia, como já realizado por Astrova

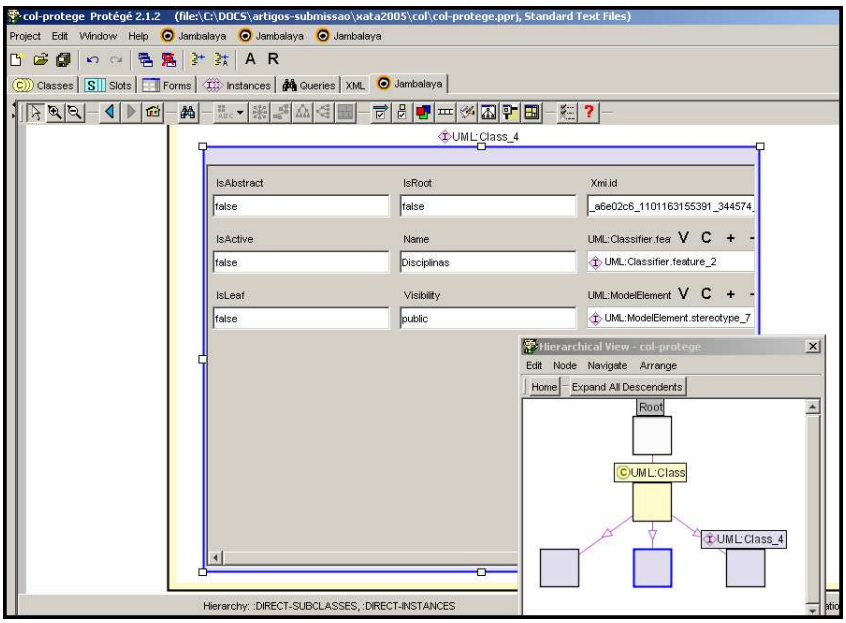

Figura 5: Subconjunto da ontologia inicial, obtido a partir do padrão XMI

Modelo Relacional	Diagrama de Classe UML	Ontologia
Tabela	Classe	Conceito
Coluna	Atributo	Atributo
Chave estrangeira	Associação	Propriedade

Tabela 1: Transformações obtidas

[1], Handshuh [12], Knublauch [16] e proposto neste artigo, pode auxiliar muito na construção de um modelo inicial de ontologia para um domínio específico. Este modelo passa a ser o ponto de convergência dos engenheiros ontológicos, permitindo o compartilhamento do conhecimento representado e a descrição dos conceitos mais comuns. Para a representação completa da ontologia, é necessário realizar a engenharia progressiva. Como futuros trabalhos, pretende-se aplicar os processos definidos, partindo de metadados existentes em ferramentas de desenvolvimento de software, como J2EE, e também utilizar o aprendizado de ontologia (*Ontology Learning*) [19], para o tratamento dos diversos documentos do sistema.

Referências

[1] I. Astrova. Reverse engineering of relational databases to ontologies. In C. Bussler, J. Davies, D. Fensel, and R. Stude, editors, *The Semantic Web: Research and Applications, First European Semantic Web Symposium (ESWS)*, pages 327–341, Heraklion, Crete, Greece, May 2004. ISBN 3-540-21999-4.

[2] K. Baclawski, M. Kokar, P. Kogut, L. Hart, J. Smith, W. Holmes, J. Letkowski, and M. Aroston. UOL: Unified ontology language. Assorted papers discussed at the DC Ontology SIG Meeting, 2002. http://www.omg.org/cgi-bin/doc?ontology/2002-11-02.

[3] T. Berners-Lee. The World Wide Web - past present and future. http://www.w3.org/2002/04/Japan/Lecture.html, 2002.

[4] E. J. Chikofsky and J. H. Cross II. Reverse engineering and design recovery: A taxonomy. *IEEE Software*, 7(1):13–17, 1990.

[5] Ó. Corcho, M. Fernández-López, A. Gómez-Pérez, and A. López-Cima. Building legal ontologies with methontology and webode. In *Law and the Semantic Web*, pages 142–157, 2003.

[6] S. Cranefield. UML and the semantic web. In *International Semantic Web Working Symposium*, 2001.

[7] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: A tool for colaborative ontology construction. volume 46, pages 707–728, June 1997.

[8] A. Gómez-Pérez and D. Manzano-Macho. A survey of ontology learning methods and techniques. http://ontoweb.aifb.uni-karlsruhe.de/Members/ruben/Deliverable, 2003.

[9] M. Grüninger and M. S. Fox. The role of competency questions in enterprise engineering. In *Workshop on Bechmarking, Theory and Practice*, Trondheim, Norway, 1994. http://www.ie.utoronto.ca/EIL/public/competency.ps.

[10] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. 43:907–928, 1995.

[11] N. Guarino. Understanding, building, and using ontologies: a commentary to using explicit ontologies. 46:293–310, 1997.

[12] S. Handschuh. KAON - the karlsruhe ontology and semantic web infrastructure. Technical report, Forschungszentrum Informatik Karlsruhe, 2001. http://kaon.semanticweb.org/papers.

[13] I. Jacobson and F. Lindstrom. Reegineering of old systems to an object-oriented architeture. In *SIGPLAN Notices*, volume 26, pages 340–350, 1991.

[14] R. Karp, V. Chaudhri, and J. Thomere. XOL: AnXML-Based Ontology Exchange Language (version 0.4). www.ai.sri.com/ pkarp/xol, August 1999.

[15] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.

[16] H. Knublauch. Ontology driven software development in the context of the semantic web: An example, scenario with protégé/owl. *1st International Workshop on the Model-Driven Semantic Web (MDSW2004)*, 2004.

[17] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[18] S. Luke and J. Heflin. SHOE 1.01. Proposed Specification - SHOE Project. Technical report, University of Maryland, 2000. http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm.

[19] A. Maedche. *Ontology learning for the Semantic Web*. Kluwer Academic Publishers, Massachusetts, 2002. 241p.

[20] OMG. Meta-object facility (MOF). http://www.omg.org/cgi-bin/apps/doc?formal/02-04-03.pdf, 2002.

[21] OMG. XML metadata interchange (XMI). http://www.omg.org/cgi-bin/doc?formal/2002-01-01, 2002.

[22] OMG. Unified modeling language (UML). http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.zip, 2003.

[23] R. S. Pressman. *Engenharia de Software*. Mcgraw-Hill Interamericana do Brasil, 2002. 5a. edição.

[24] A. Schreiber, B. Wielinga, H. Akkermans, W. van de Velde, and A. Anjewierden. CML: The CommonKADS conceptual modeling language. In S. *et al., editor,* Proc. 8th European Knowledge Acquisition Workshop (EKAW94) - A Future of Knowledge Acquisition. *Springer-Verlag, 1994. Lecture Notes in Artificial Intelligence 867.*

[25] M. Uschold and M. King. Towards a methodology for building ontologies. In IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.*