Ontologias e Ambientes de Desenvolvimento de Software Semânticos

Ricardo A. Falbo, Fabiano B. Ruy, Juliana Pezzin, Rodrigo Dal Moro

Departamento de Informática, Universidade Federal do Espírito Santo, Vitória - ES - Brasil
{falbo, fruy}@inf.ufes.br, {jpezzin, rdalmoro}@yahoo.com.br

Abstract. Software Development Environments (SDEs) provide a means to integrate developers with the software process and the supporting technology. Since during software development many information resources are produced and used, it is very important to add semantics to them in order to improve the assistance given by the environment. In this context, ontologies are a key enabling technology for Semantic SDEs (SSDEs). A SSDE can be viewed as a SDE in which part of the information handled has associated a formal meaning (semantics), augmenting its tools' ability to work in cooperation each other and with human developers. This paper discusses how ontologies are used in ODE, an Ontology-based software Development Environment, to make it a SSDE.

1 Introdução

Desde a disseminação do uso de computadores em diversos campos do conhecimento humano, a principal preocupação da área de Tecnologia de Informação tem sido prover informações para apoiar a resolução de problemas. Entretanto, como resultado deste esforço, nos últimos anos, um novo problema surgiu: o excesso de recursos de informação, associado à falta de semântica para guiar uma busca por recursos realmente relevantes para o contexto em mãos. Assim como a falta de informação constitui um problema grave, o excesso de recursos de informação também o é, já que, em última instância, pode não ser possível coletar em tempo hábil a informação necessária para apoiar a tomada de decisão durante a resolução de problemas. Este fato pode ser claramente percebido na *Web*, onde, muitas vezes, achar informação relevante pode ser uma tarefa árdua e complexa, e está relacionado, sobretudo, à falta de semântica associada aos recursos de informação.

No contexto específico da *Web*, este problema foi reconhecido e as iniciativas para tentar minimizar seus efeitos deram origem à área de pesquisa denominada *Web* Semântica (*Semantic Web*) [1]. No contexto da *Web* Semântica, busca-se tratar as informações da *Web* como uma rede de conceitos em contraposição a uma rede de documentos. A idéia é associar conhecimento do significado aos recursos da *Web*, tipicamente através da utilização de (meta) dados processáveis por máquinas. Cada conceito pode estar relacionado a outros conceitos e pode ter um grupo de recursos de informação associados. Esta rede de conceitos e recursos de informação é usada, então, na navegação na *Web* [2]. Para definir a rede de conceitos na *Web* Semântica, ontologias têm sido sistematicamente utilizadas [3].

Situação análoga ocorre nos Ambientes de Desenvolvimento de Software (ADSs). ADSs são conjuntos de ferramentas CASE integradas que facilitam a realização de atividades de engenharia de software, apoiando todo o ciclo de vida de software [4]. Uma vez que o desenvolvimento de software é um esforço criativo, complexo e coletivo, é também uma tarefa de conhecimento intenso, na qual muitos recursos de informação são produzidos e utilizados. Apesar do escopo ser menos abrangente que o escopo da *Web* Semântica, à medida que os ADSs crescem e oferecem mais funcionalidades de apoio ao desenvolvimento de software, principalmente com a incorporação de facilidades de gerência de conhecimento, problemas similares aos anteriormente descritos tornam-se preocupantes. Assim, da mesma forma que a *Web* tem avançado para *Web* Semântica, os ADSs têm de avançar para ADSs Semânticos [5] e, neste contexto, ontologias são essenciais.

Este trabalho discute como ontologias têm sido utilizadas no ambiente de desenvolvimento de software ODE (*Ontology-based software Development Environment*) [5] para torná-lo um ADS Semântico. A seção 2 discute brevemente ontologias e seu papel no estabelecimento de semântica para recursos de informação. A seção 3 apresenta sucintamente o ambiente ODE e discute como ontologias são usadas para estruturar esse ambiente e sua infra-estrutura de gerência de conhecimento. Discute-se, ainda, como ontologias são usadas para apoiar a comunicação entre agentes em ODE. A seção 4 discute alguns trabalhos relacionados e, finalmente, a seção 5 apresenta conclusões e perspectivas futuras para o desenvolvimento deste trabalho.

2 Ontologias e Semântica de Recursos de Informação

O crescimento rápido e contínuo do volume de informações torna cada vez mais difícil encontrar, organizar, acessar e manter informação requerida por usuários. Muitas vezes, pedaços importantes da informação relevante estão dispersos em diferentes recursos de informação e o meio mais utilizado para encontrá-los é através de máquinas de busca. Entretanto, as máquinas de busca tradicionais retornam listas de recursos recuperados, oferecendo pouca ou nenhuma informação sobre as relações semânticas existentes entre eles. Por conseguinte, o usuário tem de despender uma quantidade substancial de tempo acessando-os e lendo-os, para então descobrir como esses recursos de informação estão relacionados e onde eles se encaixam na estrutura geral do domínio do problema. Neste contexto, ontologias oferecem um meio de lidar com a representação de recursos de informação: o modelo de domínio descrito por uma ontologia pode ser usado como uma estrutura unificadora para dar semântica e uma representação comum à informação [3].

Ontologias têm se tornado populares, em grande parte, pelo fato de terem como objetivo promover um entendimento comum e compartilhado sobre um domínio, que pode ser comunicado entre pessoas e sistemas de aplicação [3]. Uma ontologia define um vocabulário específico usado para descrever uma certa realidade, mais um conjunto de decisões explícitas fixando de forma rigorosa o significado pretendido para o vocabulário. Uma ontologia envolve, então, um vocabulário de representação

que captura os conceitos e relações em algum domínio e um conjunto de axiomas, que restringem a sua interpretação [6].

O potencial do uso de ontologias para lidar com o problema da semântica de recursos de informação, sobretudo quando há grandes volumes de informação, tem sido largamente explorado pelas áreas de pesquisa da *Web* Semântica [1,3] e da Gerência de Conhecimento [3,7], onde esse problema é claramente crucial. Entretanto, pouco se tem explorado no campo de pesquisa dos Ambientes de Desenvolvimento de Software (ADSs), onde o problema é igualmente relevante. Durante um projeto de software, muitas informações são produzidas e requeridas e, em muitas situações, é essencial estabelecer conexões entre recursos de informação para se obter o conjunto necessário de informações para apoiar a realização de uma atividade. Assim sendo, os atuais ADSs têm de evoluir para ADSs Semânticos e ontologias desempenham um papel fundamental neste contexto. Na próxima seção, discute-se como ontologias são usadas no ambiente ODE, visando torná-lo um ADS Semântico.

3 ODE: Um ADS Baseado em Ontologias

ODE (*Ontology-based software Development Environment*) [5] é um ADS centrado em processo, que tem sua fundamentação baseada em ontologias. Em ODE, parte-se do pressuposto que, se as ferramentas de um ADS são construídas baseadas em ontologias, a integração delas pode ser facilitada, pois os conceitos envolvidos são bem definidos pelas ontologias [5]. ODE está sendo desenvolvido no Laboratório de Engenharia de Software da Universidade Federal do Espírito Santo (LabES/UFES) e sua versão atual está em vias de ser implantada em uma *software house*. ODE é implementado em Java e possui várias ferramentas, dentre elas: de apoio à definição de processos de software [8], de apoio à gerência de riscos (GeRis) [9], de apoio à documentação (XMLDoc) [10], de apoio à gerência de recursos humanos, de realização de inferências [8] e de edição de ontologias (ODEd) [11].

Dentre as ontologias que compõem a base ontológica de ODE, tem-se as ontologias de processo de software [12], de qualidade de software [13], de artefatos de software [10] e de riscos de software [9]. Essas ontologias são usadas, dentre outros, para estruturar o ambiente e sua infra-estrutura de gerência de conhecimento e para estabelecer uma forma padrão de comunicação entre os agentes que atuam no ambiente, conforme discutido nas subseções a seguir.

3.1 Derivando a Estrutura do Ambiente a partir de Ontologias

Por ser baseado em ontologias, ODE realiza grande parte de suas tarefas utilizando e respeitando os modelos e restrições impostos pelas ontologias sobre as quais se fundamenta. Assim, tem-se tentado implementar essa estreita relação entre ambiente e ontologias de uma forma natural, uma vez que há uma enorme dependência do ambiente em relação às ontologias.

Dado que ODE é desenvolvido usando a tecnologia de objetos, optou-se por realizar um mapeamento de ontologias para modelos de objetos, aplicando-se a

abordagem sistemática de derivação de infra-estruturas de objetos a partir de ontologias, descrita em [13]. Essa abordagem permite que os elementos definidos em uma ontologia (conceitos, relações, propriedades e restrições definidas como axiomas) sejam mapeados para um modelo de objetos que é, então, integrado como parte fundamental da estrutura do ambiente.

Com o intuito de manter a amarração semântica entre os objetos de ODE e agregar ontologias ao ambiente, sua arquitetura conceitual[1] foi projetada em três níveis, como mostra a figura 1, discutidos a seguir:

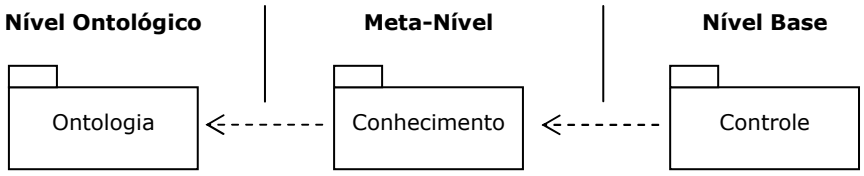

Fig. 1. Os três níveis da arquitetura conceitual de ODE.

- O Nível Ontológico, ou pacote *Ontologia*, é responsável pela descrição das ontologias em si. Nele encontram-se as classes que definem explicitamente uma ontologia e seus elementos e, portanto, o modelo de classes desse nível (figura 2) corresponde à meta-ontologia adotada no ambiente ODE. O objetivo do Nível Ontológico é registrar as ontologias em ODE, descrevendo o que é essencial para um certo domínio (compromissos ontológicos mínimos) e, portanto, não é de sua responsabilidade prover detalhes de implementação de sistemas nesse domínio. Contudo, vale ressaltar que as instâncias do nível ontológico guiam a definição das classes dos outros níveis, originando as principais classes tanto do meta-nível (ou nível de conhecimento) quanto do nível base.
- O Meta-nível, ou pacote *Conhecimento*, abriga as classes que descrevem o conhecimento em relação a um domínio de aplicação. Suas classes são derivadas das ontologias e as instâncias dessas classes atuam no papel de conhecimento sobre os objetos do nível base. A derivação das classes do nível de Conhecimento é feita diretamente a partir do nível ontológico, isto é, suas classes correspondem a conceitos descritos como instâncias da classe *Conceito* (vide figura 2) no pacote *Ontologia*. Elas constituem o conhecimento do ambiente, que pode ser utilizado tanto pelo ambiente quanto pelas ferramentas que o compõem.
- O Nível Base, ou de Aplicação (pacote *Controle*), define as classes responsáveis por implementar as aplicações no contexto do ambiente (funcionalidades da infra-estrutura do ambiente e suas ferramentas). Essas classes são também derivadas das ontologias, mas tipicamente incorporam detalhes não descritos por elas, necessários para implementar as aplicações do ambiente. Por não se ter uma definição de características particulares para um sistema no nível ontológico

[1] Está-se usando o termo "arquitetura conceitual" para indicar uma decomposição de alto nível dos pacotes do ambiente, em contraste à arquitetura de software em camadas utilizada para implementá-lo efetivamente.

(compromissos ontológicos mínimos), muitas vezes é necessária a criação de novas classes, associações, atributos e operações com o intuito de se tratar decisões específicas do nível de aplicação.

Essa divisão arquitetural facilita o estabelecimento de uma correlação entre objetos dos diferentes níveis de ODE, permitindo anotar os objetos com informação semântica, dada efetivamente pelas ontologias. A seguir, discute-se como a estrutura do ambiente é desenvolvida tomando por base essa arquitetura conceitual e o suporte de ferramentas oferecido para realizar essa tarefa.

ODEd: O Editor de Ontologias de ODE

Conforme discutido anteriormente, o nível Ontológico descreve as ontologias em ODE e, para tal, o ambiente disponibiliza o editor de ontologias ODEd (*ODE's ontology Editor*) [11]. ODEd é uma ferramenta gráfica que apóia o desenvolvimento de ontologias através da definição de seus conceitos, relações e propriedades, e que permite a definição de axiomas e avaliação de ontologias, através de um editor de axiomas integrado [14]. ODEd utiliza o modelo de meta-ontologia mostrado na figura 2 para construir e registrar as ontologias do ambiente no nível ontológico.

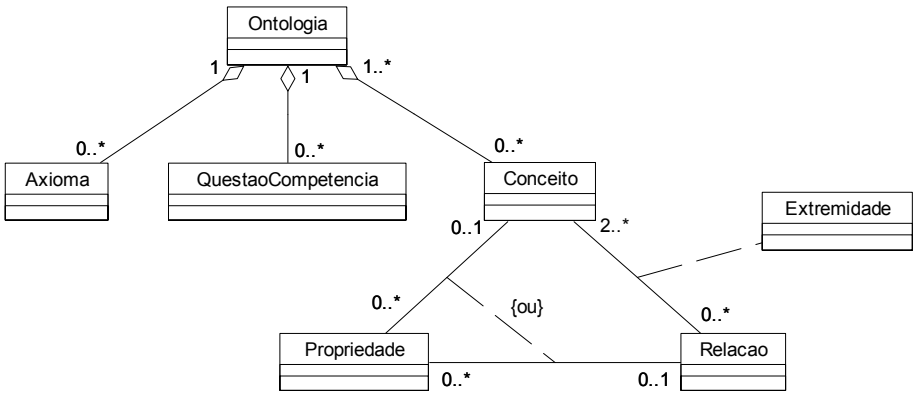

Fig. 2. Modelo de Classes da Meta-Ontologia.

Neste nível, os objetos criados a partir da edição de uma ontologia representam os elementos pertencentes às ontologias de ODE. Tomando por base a ontologia de processo de software [12], parcialmente apresentada na figura 3, tem-se a própria *Ontologia de Processo de Software* representada como uma instância da classe `Ontologia`; *Projeto*, *Processo*, *Atividade*, *Recurso*, *Recurso Humano*, *Ferramenta de Software*, *Procedimento* e *Artefato*, por sua vez, são instâncias da classe `Conceito`; *posse*, *sub-atividade*, *insumo*, *produto*, *possível adoção* e *uso* são instâncias da classe `Relacao`; e, finalmente, existem, ainda, algumas questões de competências e alguns axiomas (não exibidos graficamente), relativos a *sub-atividade* e *uso*, por exemplo, que são tratados como instâncias das classes `QuestaoCompetencia` e `Axioma`, respectivamente.

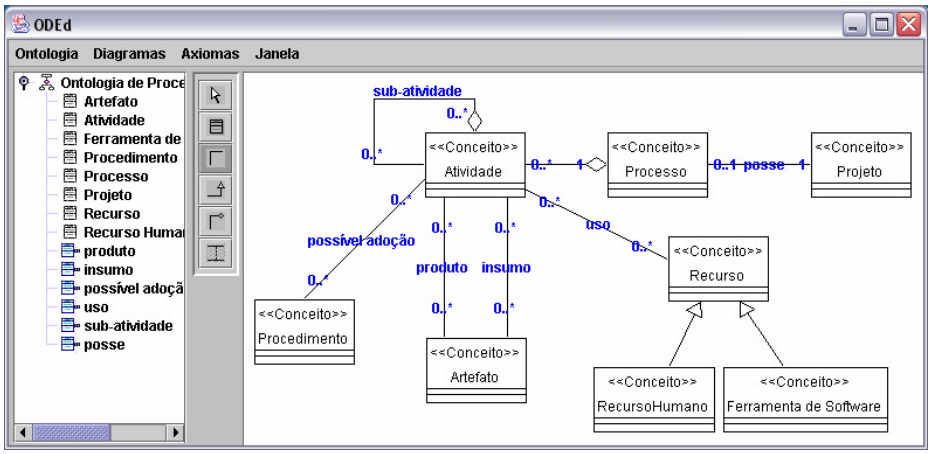

Fig. 3. Construção parcial da Ontologia de Processo de Software em ODEd.

Derivação das Classes dos Níveis de Conhecimento e de Aplicação

Uma vez definida uma ontologia no nível ontológico, é possível derivar os modelos de objetos que compõem os outros dois níveis da arquitetura conceitual de ODE. Neste momento a abordagem de derivação proposta em [13] é utilizada. Conceitos e relações são naturalmente mapeados em classes e associações em um modelo de objetos. Propriedades de conceitos e relações são mapeadas em atributos das classes originadas a partir do respectivo conceito/relação. Axiomas, por sua vez, são mapeados em métodos [13]. Esta derivação é parcialmente apoiada por ODEd, que gera uma infra-estrutura básica de objetos que posteriormente é alterada pelos desenvolvedores do ambiente.

Como a arquitetura conceitual de ODE possui, além do nível ontológico, outros dois níveis, o processo de derivação de ontologias em modelos de objetos não dá origem a apenas um modelo de objetos, mas a dois. Dessa forma, durante o processo de derivação, um determinado conceito pode derivar uma classe somente no nível de conhecimento, somente no nível base, em ambos os níveis, ou mesmo em nenhum deles, conforme discutido a seguir.

- **Nenhuma classe é criada.**

Apesar de muito pouco freqüente, alguns conceitos podem não ser necessários fora do escopo da ontologia. Eles podem ter sido definidos somente para esclarecer algum aspecto da ontologia, mas quando observados do ponto de vista sistêmico, eles podem não ter um papel relevante em um modelo de objetos. Neste caso, não é necessário criar classe alguma para o conceito.

- **São criadas classes em ambos os níveis: de conhecimento e de aplicação.**

Muitas vezes, um conceito de uma ontologia é necessário nos dois níveis: no de conhecimento, determinando o tipo dos objetos concretos do mundo real, e no nível de aplicação, representando os próprios objetos do mundo real. Nessa situação, os objetos do nível de conhecimento são utilizados para descrever os objetos do nível de aplicação, através de uma referência (atributo), como mostra a figura 4.

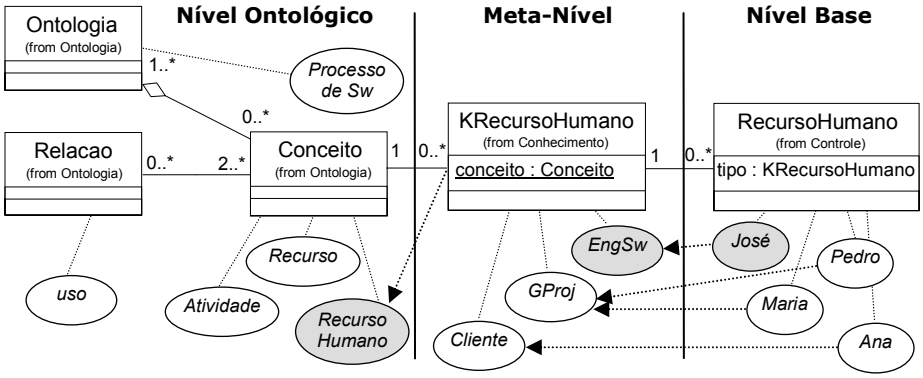

Fig. 4. Derivação de Classes nos Níveis de Conhecimento e de Aplicação.

No exemplo da figura 4, o conceito *Recurso Humano*, instância da classe `Conceito` no nível ontológico, dá origem a duas classes: `KRecursoHumano`[2] no nível de conhecimento e `RecursoHumano` no nível base. A primeira representa os "tipos" de recursos humanos potencialmente importantes em um processo de desenvolvimento de software, tais como *Engenheiro de Software*, *Gerente de Projeto, Cliente* etc. Essas instâncias do nível de conhecimento são utilizadas para classificar os objetos concretos do nível base (*José*, *Maria*, *Pedro*, *Ana* etc). Dessa maneira, pode-se dizer que *José* é um *Engenheiro de Software* que, por sua vez, é um *Recurso Humano*.

De maneira análoga, o conceito *Atividade* dá origem às classes `KAtividade` no nível de conhecimento e `Atividade` no nível base, e pode-se utilizar essas classes para descrever, respectivamente, tipos de atividades no desenvolvimento de software (por exemplo, *Planejamento*, *Especificação de Requisitos* etc) e atividades concretas realizadas no contexto de um projeto específico (por exemplo, *Planejamento Inicial do Projeto X*, *Especificação de Requisitos Preliminar do Projeto X* etc). Neste caso, uma vez que a classe `Atividade` é anotada com uma referência a uma instância da classe `KAtividade`, pode-se dizer que *Planejamento Inicial do Projeto X* é uma atividade do tipo *Planejamento*.

Finalmente, como a ontologia de processo de software (figura 3) define que recursos podem ser usados por atividades, pode-se descrever no nível de conhecimento que atividades do tipo *Especificação de Requisitos* requerem recursos

[2] Em ODE, as classes de conhecimento (meta-nível) são nomeadas com o prefixo K, de *Knowledge*. Assim, todas as classes iniciadas com K são subclasses da classe Conhecimento.

humanos do tipo *Engenheiro de Software*. Na alocação de recursos humanos para o *Projeto X*, essa informação é utilizada para apontar que *José* pode ser alocado à atividade *Especificação de Requisitos Preliminar do Projeto X*, já que ele é um recurso humano compatível com as necessidades dessa atividade. Desta forma, o nível de conhecimento pode ser usado, também, para guiar a realização de atividades do nível base.

- **É criada uma classe somente, ou no nível de conhecimento ou no nível base.**

Esta situação acontece quando o conceito só possui um nível relevante de instâncias. Por exemplo, seja o conceito *Procedimento*. Instâncias desse conceito incluem, dentre outros, *Análise Estruturada*, *Análise Orientada a Objetos*, *Inspeção de Código* etc. Essas instâncias são suficientes para ambos os níveis de conhecimento e de aplicação. Tomando os exemplos de atividades mencionados anteriormente, é possível dizer, no nível de conhecimento, que a atividade de *Especificação de Requisitos* pode adotar como procedimento para sua execução o método da *Análise Orientada a Objetos*. Por outro lado, pode-se dizer, também no nível de aplicação, que a atividade *Especificação de Requisitos Preliminar do Projeto X* é conduzida seguindo o método da *Análise Orientada a Objetos*. Ou seja, apenas uma classe é suficiente para tratar os dois níveis. Neste caso, como a classe derivada é importante para o nível de conhecimento, isto é, o que está sendo descrito por ela é, de fato, um conhecimento sobre os procedimentos que podem ser adotados no desenvolvimento de software, ela é criada no nível de conhecimento (`KProcedimento`). Como o nível base tem acesso ao nível de conhecimento, ele também pode utilizá-la quando necessário.

Em algumas situações, contudo, certos conceitos originam classes somente no nível base. Isso ocorre quando o conceito só possui um nível relevante de instâncias e essas instâncias são importantes apenas no nível de aplicação, como é o caso em que um tipo não é necessário, mas os objetos concretos do mundo real o são. Tomando o exemplo da ontologia de processo de software, este é o caso do conceito *Projeto*. Quando se fala em um projeto, está se referindo a um projeto específico tal como o *Projeto X*, que possui as atividades *Planejamento Inicial do Projeto X*, *Especificação de Requisitos Preliminar do Projeto X* etc. e que tem como membros de sua equipe os recursos humanos *José*, *Maria*, *Pedro*, *Ana* etc. Assim, esse conceito é derivado somente para o nível base, dando origem à classe `Projeto`.

Observando-se a dependência entre os níveis arquiteturais de ODE (figura 1), verifica-se que o nível base depende do nível de conhecimento, que, por sua vez, depende do nível ontológico. As classes de cada nível são desenvolvidas respeitando essa dependência, ou seja, as classes do nível base possuem uma associação com sua correspondente no nível de conhecimento. E as classes do nível de conhecimento estão associadas ao seu conceito em uma ontologia (vide figura 4).

Esta é uma forma de anotar os conceitos das ontologias nos objetos dos outros dois níveis. Essa anotação permite que os objetos do ambiente com origem em algum conceito de uma ontologia possam facilmente identificar essa origem. Dessa forma, a navegação entre os níveis é simplificada e diversas tarefas do ambiente (definição de processos, alocação de recursos, gerência de riscos, configuração do ambiente etc) podem ser apoiadas e validadas utilizando uma perspectiva ontológica.

Essa correspondência com as ontologias pode ser ainda mais amplamente explorada através do uso de inferências. A ferramenta de realização de inferências [8] integrada a ODE permite a definição de regras em Prolog a partir dos axiomas ontológicos. O processamento dessas regras em conjunto com objetos do sistema por uma máquina de inferência oferece às ferramentas do ambiente um suporte inteligente na realização de suas tarefas.

Como exemplo, seja o axioma da ontologia de processos de software que diz que "se um recurso R é usado por uma atividade A e A é sub-atividade de outra atividade B, então R também é usado por B". Esse axioma pode ser convertido em uma regra da ferramenta de inferências e, posteriormente, utilizado na ferramenta de definição de processos para apresentar uma sugestão mais ampla de recursos que podem ser usados por uma determinada atividade.

Dessa forma, em diversas tarefas do ambiente, os objetos dos níveis de aplicação ou mesmo de conhecimento podem ter sua anotação ontológica utilizada para realização de tarefas de forma mais inteligente e consistente segundo uma visão ontológica.

3.2 A Infra-estrutura de Gerência de Conhecimento de ODE

Tendo em vista a importância de se integrar e gerenciar o conhecimento adquirido durante o desenvolvimento dos projetos de software, foi desenvolvida uma infra-estrutura para apoiar a gerência de conhecimento em ODE [15]. Essa infra-estrutura, como mostra a figura 5, é composta por uma memória organizacional e por um conjunto de serviços de gerência de conhecimento, que incluem: (i) *criação e captura de conhecimento*, responsável por oferecer mecanismos para obtenção e armazenamento do conhecimento; (ii) *recuperação e acesso ao conhecimento*, responsável por oferecer mecanismos de busca dos itens de conhecimento armazenados na memória organizacional; (iii) *uso do conhecimento*, responsável por apoiar o reuso do conhecimento existente por parte do usuário e oferecer mecanismos de realimentação sobre a utilidade do conhecimento apresentado; (iv) *manutenção do conhecimento*, responsável pelo gerenciamento dos repositórios de conhecimento, tomando por base o *feedback* dos usuários; e (v) *disseminação de conhecimento*, serviço pró-ativo realizado por agentes de software com o intuito de disponibilizar aos usuários itens de conhecimento potencialmente úteis a uma dada tarefa em que os mesmos estejam trabalhando.

No caso dos serviços de disseminação, agentes de software monitoram as ações dos usuários e, percebendo a execução de tarefas numa dada ferramenta, agem de forma autônoma, identificando as necessidades de conhecimento naquele contexto. Dessa forma, os agentes recuperam itens de conhecimento da memória organizacional e os exibem pró-ativamente ao usuário.

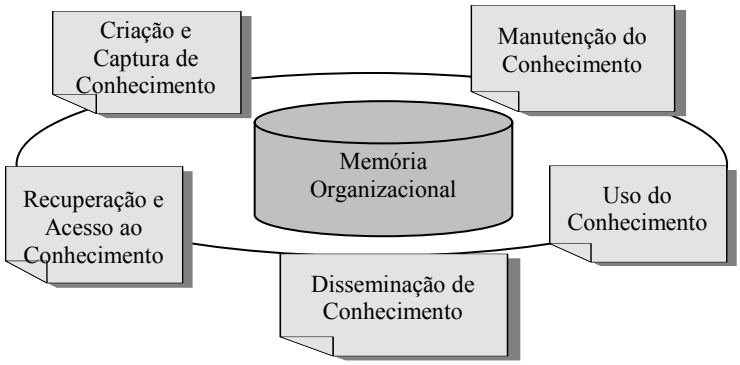

Fig. 5. A Infra-estrutura de Gerência de Conhecimento de ODE.

Conforme apontado em [7], ontologias são a cola que mantém ligadas as atividades da gerência de conhecimento, definindo um vocabulário comum a ser utilizado pelo sistema de gerência de conhecimento e, por conseguinte, facilitando a comunicação, integração, busca, armazenamento e representação do conhecimento.Com base nessa premissa, a estrutura da memória organizacional de ODE é definida, também, fortemente apoiada em ontologias, como mostra a figura 6. Nela, os itens de conhecimento podem ser classificados em itens de conhecimento formais e informais. Os itens de conhecimento formais são os diversos tipos de artefatos gerados pelas ferramentas do ambiente. Já os itens de conhecimento informais compreendem, atualmente, lições aprendidas e pacotes de mensagens.

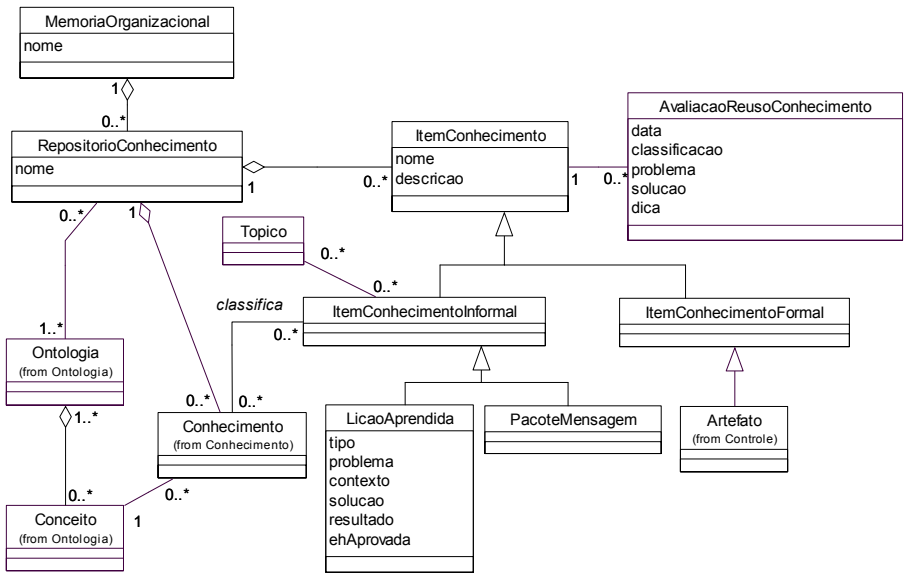

Fig. 6. Modelo da Infra-estrutura de Gerência de Conhecimento de ODE.

Tanto os itens de conhecimento quanto as instâncias de ontologias (objetos da classe `Conhecimento`) compõem os repositórios de conhecimento do ambiente, que formam a memória organizacional. De fato, os objetos da classe `Conhecimento` podem ser vistos como um tipo de item de conhecimento formal, já que são itens de conhecimento utilizados em várias situações no ambiente e são formalmente definidos a partir de ontologias. Entretanto, como mostra a figura 6, optou-se por não colocá-los como sub-classe de `ItemConhecimentoFormal`, uma vez que parte dos serviços da gerência de conhecimento (tal como a caracterização de reuso) não se aplica a esses objetos.

Pode-se notar que os itens de conhecimento, formais ou informais, são anotados segundo conceitos de ontologias. Considerando-se os itens de conhecimento formais, os artefatos, que fazem parte do nível base, eles possuem uma correspondência com a classe `KArtefato` do nível de conhecimento, que, por sua vez, está ligada ao conceito *Artefato* no nível ontológico. Já os itens de conhecimento informais (lições aprendidas e pacotes de mensagens) são classificados usando instâncias da classe `Conhecimento`, correlacionando-se, assim, indiretamente a conceitos das ontologias.

O esquema de anotação ontológica facilita a oferta de diversos serviços da gerência de conhecimento de ODE, tal como o serviço de busca, ilustrado na figura 7. A busca pode ser aplicada tanto a itens de conhecimento formais quanto informais. Como exemplo, a busca de uma lição aprendida pode ter como critérios de filtro: tópicos relacionados, palavras-chave referentes à sua descrição, o projeto ao qual a lição pertence ou ainda o tipo da mesma. Contudo, um ponto importante da busca é a possibilidade de recuperar itens de acordo com sua classificação segundo instâncias de ontologias dos repositórios de conhecimento de ODE. No exemplo da figura 7, deseja-se recuperar lições aprendidas que, entre outros critérios, tenham sido classificadas segundo duas instâncias da ontologia de processo, neste caso, instâncias do conceito de atividade. Assim, dois objetos da classe `KAtividade` são utilizados na recuperação das lições aprendidas, utilizando, portanto, as anotações ontológicas.

3.3 Ontologias na Comunicação entre Agentes de ODE

Por serem sistemas de software complexos, Ambientes de Desenvolvimento de Software (ADSs) são potenciais beneficiários da tecnologia de agentes. Como um exemplo da utilidade dessa tecnologia, pode-se citar a disseminação de conhecimento na gerência de conhecimento, conforme discutido na seção anterior. Assim, em ODE, agentes são utilizados para aperfeiçoar algumas das funcionalidades do ambiente e uma infra-estrutura para apoiar a construção desses agentes, denominada AgeODE [16] foi desenvolvida.

Dentre os agentes desenvolvidos em ODE, podem ser citados o agente assistente pessoal (*AgAssistentePessoal*) e o agente gerente de projeto (*AgGerenteProjeto*) [16], cujo objetivo é acompanhar o usuário que está utilizando o ambiente. O agente assistente pessoal acompanha o usuário de ODE desde o momento em que ele acessa o ambiente até o momento em que ele sai, dando-lhe sugestões, avisos etc. Ao iniciar o ambiente, o agente assistente pessoal é ativado e quando um usuário se identifica em ODE, o agente captura quem é esse usuário. Isso também ocorre quando o usuário seleciona um projeto para trabalhar.

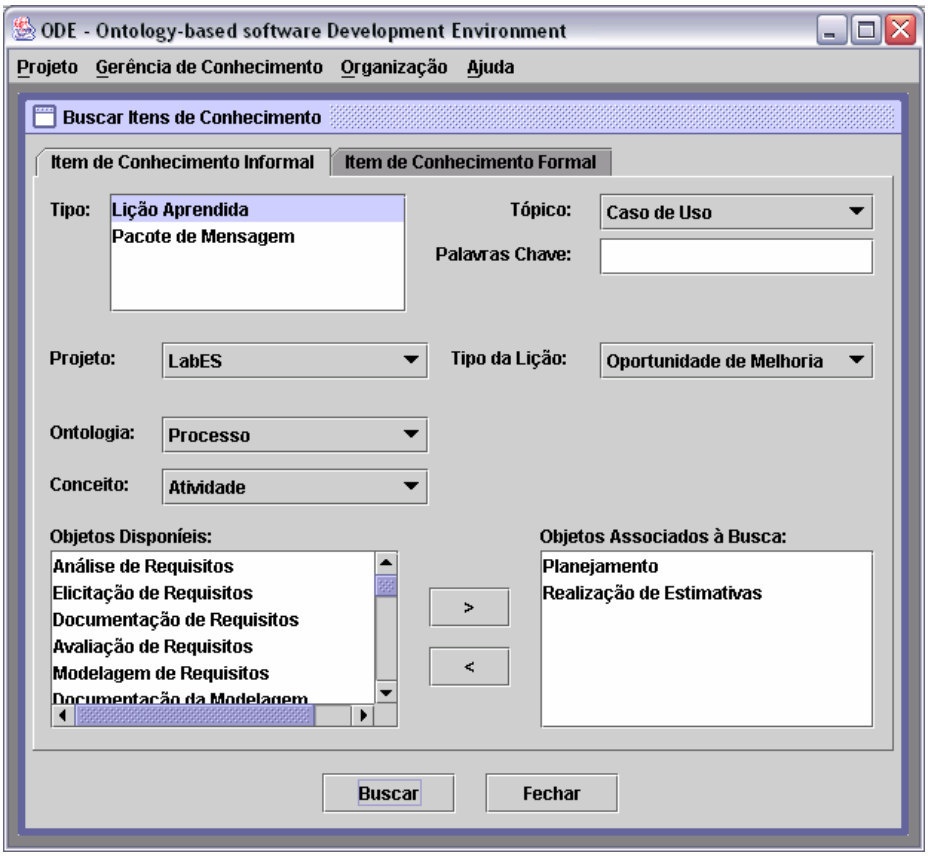

Fig. 7. Serviço de Busca da Gerência de Conhecimento de ODE.

Ao capturar o projeto, o agente assistente pessoal pode verificar se o usuário é, ou não, gerente do projeto recém selecionado. Caso o usuário seja o gerente desse projeto, o agente assistente pessoal inicia o agente gerente de projeto, que, ao ser ativado, envia uma mensagem ao agente assistente pessoal perguntando o usuário e o projeto. O agente gerente de projeto atua auxiliando o gerente de projeto no acompanhamento do projeto, indicando, por exemplo, quais atividades estão em atraso no projeto.

Como pode ser visto pelo exemplo comentado acima, agentes devem ter a habilidade de se comunicar. Essa habilidade é parte percepção (o recebimento de mensagens) e parte ação (o envio de mensagens).

A comunicação entre os agentes de ODE é feita usando KQML *(Knowledge Query and Manipulation Languag*e) [17] e é intermediada por um agente roteador, que distribui as mensagens aos demais agentes. As primitivas de KQML, chamadas de *performativas*, definem as ações admissíveis que os agentes podem tentar na comunicação com outros agentes. A sintaxe de KQML é baseada em uma lista de argumentos, na qual o elemento inicial é a performativa. Os elementos restantes são os parâmetros da performativa, representados por pares (palavra-chave,valor).

Novamente, ontologias são bastante úteis, agora para apoiar a comunicação entre os agentes. O projeto de um sistema multiagente requer a definição de um modelo do ambiente no qual o agente atua, para que este possa conversar sobre ele. Esse modelo pode ser exatamente uma ontologia. Desta forma, para que um agente consiga se comunicar com outro agente, eles devem conhecer a(s) mesma(s) ontologia(s).

As mensagens entre agentes de ODE obrigatoriamente têm de descrever o parâmetro KQML *:ontology*, informando as ontologias envolvidas na comunicação. Assim, é possível verificar se a comunicação pode ser bem sucedida. A seguir, para ilustrar o conteúdo das mensagens enviadas entre os agentes de ODE, são mostradas as mensagens trocadas entre os agentes assistente pessoal e gerente de projeto no contexto discutido anteriormente. Para maiores detalhes, vide [16].

Ao ser iniciado, *AgGerenteProjeto* pergunta a *AgAssistentePessoal* qual o projeto aberto. Esta mensagem é escrita em KQML conforme a seguir:

```
ask-one
   :sender AgGerenteProjeto
   :receiver AgAssistentePessoal
   :language XML
   :ontology Processo
   :content <Ode.Controle.Cdp.Projeto/>
```

Na seqüência, *AgAssistentePessoal* responde informando o projeto, enviando a seguinte mensagem, onde `id="2:101"` representa o identificador do projeto aberto.

```
reply
   :sender AgAssistentePessoal
   :receiver AgGerenteProjeto
   :language XML
   :ontology Processo
   :content <Ode.Controle.Cdp.Projeto id="2:101"/>
```

O conteúdo das mensagens (*:content*) é escrito em XML. No caso das mensagens requisitando objetos, como a mensagem do agente gerente de projeto, deve-se descrever o caminho completo da classe a que pertencem. Já no caso das mensagens informando objetos, como a resposta do agente assistente pessoal, o conteúdo informa o identificador do objeto correspondente, de modo que o agente, ao recebê-la, possa interpretar a mensagem e recuperar o objeto em questão por meio de seu identificador.

4 Trabalhos Relacionados

Não são muitos os trabalhos que têm explorado o uso sistemático de ontologias para estabelecer a semântica de itens de conhecimento em Ambientes de Desenvolvimento de Software (ADSs). Dentre os poucos trabalhos que exploram o uso de ontologias em ADSs, vale destacar a Estação TABA [18]. Nesse ambiente, assim como em ODE, ontologias são usadas para estruturar o ambiente e para apoiar a gerência de

conhecimento. Entretanto, na Estação TABA, o uso de ontologias aparenta ser mais superficial, não explorando alguns aspectos considerados em ODE, tais como a incorporação de restrições descritas na forma de axiomas e capacidades de inferência [8]. Este último aspecto, em especial, tem sido foco de pesquisas atuais em ODE [14].

No que concerne à forma como ODE utiliza ontologias, esta foi definida com base em diversos trabalhos realizados nessa área de estudo. A idéia de estruturar a arquitetura conceitual de ODE em três níveis, por exemplo, foi fortemente apoiada nos trabalhos de Guarino [6], Borst [19] e Valente et al. [20]. Valente et al. [20], por exemplo, argumentam que ontologias de domínio devem ter como objetivo a especificação das *categorias básicas* de conhecimento do domínio, que devem ser vistas como tipos de conhecimento. Guarino [6], por sua vez, aponta que não é objetivo de uma ontologia de domínio descrever todo o conhecimento a ser codificado em uma base de conhecimento. Algum conhecimento empírico, compilado ou prático, que é dependente da tarefa ou aplicação particular, pode encontrar lugar apenas em uma ontologia de aplicação. O conhecimento desempenha papéis na resolução de problemas e, uma vez que muitos desses papéis são dinamicamente atribuídos, não podem fazer parte de uma ontologia de domínio. Assim, dentro de uma base de conhecimento, é possível distinguir dois componentes: a ontologia (contendo o conhecimento do domínio) e o conhecimento específico, contendo o conhecimento dependente da aplicação. Essas idéias são claramente exploradas na estruturação de ODE, onde o meta-nível contém apenas conhecimento descrito através de ontologias, enquanto o nível base adiciona conhecimento específico para apoiar a resolução de problemas no contexto de aplicações (ferramentas) específicas de ODE.

No que concerne ao apoio à construção de ontologias, provido em ODE através de ODEd (ODE's ontology editor) [11], existem diversos editores de ontologias apresentados na literatura, tais como Protege-2000 [21], OntoEdit [22], WebODE [23] e OilEd [24], que têm funcionalidades similares, mas, de maneira geral, enfocam o suporte à construção de ontologias no contexto da *Web* Semântica. Nenhum deles é voltado especificamente para o contexto de ADSs Semânticos, como é o caso de ODEd. Assim, uma característica marcante de ODEd é apoiar um processo completo de engenharia de domínio baseada em ontologias, que envolve a construção de ontologias e a derivação de infra-estruturas de objetos a partir delas. Uma comparação mais completa entre ODEd e os editores de ontologias citados acima pode ser encontrada em [11].

5 Conclusões

Durante o desenvolvimento de software muitos recursos de informação são tratados e, em muitas situações, é essencial estabelecer conexões entre eles para se obter o conjunto necessário para apoiar a realização de uma atividade. Portanto, é importante registrar a semântica dos itens nos repositórios dos Ambientes de Desenvolvimento de Software (ADSs), de modo a evoluí-los para ADSs Semânticos. Neste contexto, ontologias desempenham um importante papel.

Uma vez que não são muitos os trabalhos que têm explorado o uso sistemático de ontologias para estabelecer a semântica de itens de conhecimento em ADSs, trabalhos

feitos em outras campos de estudo, tal como a *Web* Semântica, podem servir, por meio de analogia, de guia para avanços na área de ADSs Semânticos.

Observando a literatura voltada para a *Web* Semântica, pode-se perceber que alguns aspectos têm merecido bastante atenção, tais como: (i) linguagens para representação de ontologias [3, 25, 26], (ii) uso de XML e anotações ontológicas em documentos [3, 25, 26, 27], (iii) a forte relação entre *Web* Semântica e Gerência de Conhecimento [3, 26], (iv) capacidades de inferência e ontologias [3, 25], (v) evolução de ontologias [3] e (vi) visualização de conteúdo baseada em ontologias [27]. Assim, seguindo a linha de inspeção dos trabalhos feitos na área da *Web* Semântica e sua adaptação para o contexto de ADSs Semânticos, esses devem ser pontos a serem explorados em pesquisas futuras no âmbito do ambiente ODE. De fato, grande parte deles (itens de i a iv) já tem sido objeto de estudo no Projeto ODE e espera-se avançar nesses estudos tomando por base experiências dessa outra área de estudos.

Agradecimentos

Este trabalho foi realizado com o apoio do CNPq e da CAPES, entidades do Governo Brasileiro dedicadas ao desenvolvimento científico e tecnológico.

Referências

1. T. Berners-Lee, J. Hendler, O. Lassila, “The Semantic Web”, Scientific American, May 2001.
2. G.R. Librelotto, J.C. Ramalho, P.R. Henriques, “TM-Builder: Um Construtor de Ontologias Baseado em Topic Maps”, XXIX Conferencia Latinoamericana de Informática, 2003.
3. J. Davies, D. Fensel, F. van Harmelen, “Towards The Semantic Web: Ontology-Driven Knowledge Management”, John Wiley & Sons Ltd, 2003.
4. W. Harrison, H. Ossher, P. Tarr, “Software Engineering Tools and Environments: A Roadmap”, in Proc. of the Future of Software Engineering, ICSE’2000, Ireland, 2000.
5. R.A. Falbo, A.C.C. Natali, P.G. Mian, G. Bertollo, F.B. Ruy, “ODE: Ontology-based software Development Environment”, IX Congreso Argentino de Ciencias de la Computación, p. 1124-1135, La Plata, Argentina, Outubro 2003.
6. N. Guarino, “Formal Ontology and Information Systems”. In: Proceedings of the First Int. Conference on Formal Ontology in Information Systems, Trento, Italy, June 1998.
7. S. Staab, R. Studer, H.P. Schnurr, Y.Sure, “Knowledge Processes and Ontologies”, IEEE Intelligent Systems, vol. 16, No. 1, January/February, 2001.
8. F. Ruy, G. Bertollo, R.A. Falbo, “Knowledge-based Support to Process Integration in ODE”. Clei Electronic Journal, Volume 7, Number 1, June 2004.
9. R.A. Falbo, F.B. Ruy, G. Bertollo, D.F. Togneri, “Learning How to Manage Risks Using Organizational Knowledge”, Proceedings of the 6th International Workshop on Learning Software Organization, LSO’2004, pp. 7-18, Banff, Canada, July 2004.
10. V.B. Nunes, A.O. Soares, R.A. Falbo, “Apoio à Documentação em um Ambiente de Desenvolvimento de Software, VII Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software, IDEAS’2004, pp 50-55, Arequipa, Peru, Maio 2004.
11. P.G. Mian, R.A. Falbo, “Supporting Ontology Development with ODEd”, Journal of the Brazilian Computer Science, vol. 9, no. 2, pp 57-76, November 2003.

12. R.A. Falbo, C.S. Menezes, A.R.C. Rocha, “A Systematic Approach for Building Ontologies”. Proceedings of the 6th Ibero-American Conference on Artificial Intelligence, Lisbon, Portugal, Lecture Notes in Computer Science, vol. 1484, 1998.
13. R.A. Falbo, G. Guizzardi, K.C. Duarte, “An Ontological Approach to Domain Engineering”. Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, SEKE'2002, pp. 351- 358, Ischia, Italy, 2002.
14. V.E.S. Souza, R.A. Falbo, “Construindo Axiomas e Avaliando Ontologias em ODEd”, Anais da X Sessão de Ferramentas do Simpósio Brasileiro de Engenharia de Software, p.7-12, Manaus, Brasil, Outubro 2003.
15. A.C.C. Natali, R.A. Falbo, “Gerência de Conhecimento em ODE”, Anais do XVII Simpósio Brasileiro de Engenharia de Software, p. 270-285, Manaus, Brasil, Outubro 2003.
16. J. Pezzin, R.A. Falbo, “AgeODE: Uma Infra-estrutura para Apoiar a Construção de Agentes para Atuarem no Ambiente de Desenvolvimento de Software ODE”, 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering, JIISIC’2004, Madrid, Spain, November 2004.
17. T. Finin, Y. Labrou, J. Mayfield, “KQML as an agent communication language”. September 1995. Disponível em http://www.cs.umbc.edu/~finin/papers/
18. G. Santos, K. Villela, L. Schnaider, A.R. Rocha, G.H. Travassos, “Building Ontology-based Tools for a Software Development Environment”, Proc. of the 6th International Workshop on Learning Software Organization, LSO’2004, pp. 19-30, Banff, Canada, July 2004.
19. W.N. Borst, *Construction of Engineering Ontologies*. Centre for Telematica and Information Technology, University of Tweenty, Enschede, The Netherlands, 1997.
20. A. Valente, J. Breuker, “Ontological Engineering with Principled Core Ontologies”. Ontological Engineering - Working Notes, Stanford, California, March l997.
21. N.F. Noy, R.W. Fergerson, M.A. Musen, “The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility”. In: Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW’00). Juan-Les-Pins, France. (Lecture Notes in Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp 17-32, 2000.
22. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, D. Wenke, “OntoEdit: Collaborative Ontology Engineering for the Semantic Web”. In: Horrocks I, Hendler JA (eds) First International Semantic Web Conference (ISWC’02). Sardinia, Italy. (Lecture Notes in Computer Science LNCS 2342) Springer-Verlag, Berlin, Germany, pp 221-235, 2002.
23. J.C. Arpírez, O. Corcho, M. Fernández-López, A. Gómez-Pérez, “WebODE in a nutshell”, AI Magazine, 24(3), pp. 37-47, 2003.
24. S. Bechhofer, I. Horrocks, C. Goble, R. Stevens. “OilEd: a Reason-able Ontology Editor for the Semantic Web”. In *Working Notes of the 14th International Workshop on Description Logics (DL-2001)*, p.1-9, Stanford, EUA, August 2001.
25. G. Antoniou, F. van Harmelen. A Semantic Web Primer. The MIT Press, Cambridge, Massachusetts, 2004.
26. M.C. Daconta, L.J. Obrst, K.T. Smith. The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management. Wiley Publishing, Inc., Indianapolis, 2003.
27. V. Geroimenko, C. Chen (Eds). Visualizing the Semantic Web: XML-based Internet and Information Visualization. Springer-Verlag London, 2003.