Orientação a Domínio em ODE

Paula Gomes Mian, Ana Candida Cruz Natali,
Alessandra Lopes de Carvalho e Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo, Departamento de Informática,
Vitória, Brasil, 29060-900
{pgmian, anatali, alopes, falbo}@inf.ufes.br

Resumo

Uma das grandes dificuldades no desenvolvimento de software é que, muitas vezes, os desenvolvedores não estão familiarizados com o domínio para o qual o software está sendo desenvolvido. A necessidade de apoiar o desenvolvimento de software em domínio específicos tem sido endereçada tanto por Ambientes de Desenvolvimento de Software Orientados a Domínio (ADSODs), como pelo uso de gerência de conhecimento. Em ambos os casos, ontologias têm sido utilizadas na aquisição de conhecimento do domínio. Entretanto, a construção de ontologias não é uma tarefa simples, sendo necessário prover apoio automatizado para seu desenvolvimento. Este artigo discute o uso de ontologias e gerência de conhecimento para apoiar a orientação a domínio no ambiente ODE (*Ontology-based software Development Environment*) e apresenta ODEd, uma ferramenta de suporte à construção de ontologias que apóia a definição de conceitos e relações utilizando representações gráficas, além de promover a geração automática de certos tipos de axiomas e de *frameworks* de objetos a partir das ontologias.

Palavras-chave: Ambientes de Desenvolvimento de Software, Ambientes de Desenvolvimento de Software Orientados a Domínio, Ontologias, Gerência de Conhecimento.

Abstract

One great difficulty in software development is that, many times, developers do not know or are not familiarized with the domain in which the software is being developed. The need to support domain-oriented software development has been addressed by Domain Oriented Software Engineering Environments (DOSEEs) and Knowledge Management. In both cases, ontologies have been used for knowledge acquisition regarding the domain. However, ontology construction is not a simple task. So, it is necessary to provide tools that support ontology development. This paper discusses the use of ontologies and knowledge management to support domain-oriented software development in ODE - Ontology-based software Development Environment. It also presents an ontology editor that supports the definition of concepts and relations using graphic representations, besides promoting automatic generation of some classes of axioms and derivation of object frameworks from ontologies.

Keywords: Software Engineering Environments, Domain Oriented Software Engineering Environments, Ontology, Knowledge Management.

1 Introdução

A qualidade e produtividade de software podem ser melhoradas pelo uso de Ambientes de Desenvolvimento Software (ADSs). Estas ferramentas automatizam várias tarefas do processo de desenvolvimento de software e tornam mais fácil o seu controle. Entretanto, em muitos casos, os desenvolvedores de software não conhecem ou não estão familiarizados com o domínio para o qual devem desenvolver um determinado sistema. Ambientes de Desenvolvimento de Software Orientados a Domínio visam organizar o conhecimento sobre um domínio de aplicação, de modo a facilitar o entendimento do problema durante o desenvolvimento de sistemas. Dessa forma, o aspecto central dos ADSODs é a introdução e o uso do conhecimento do domínio no ADS [1].

Este conhecimento obtido no desenvolvimento de software é o recurso mais importante de uma organização e o seu uso promove um aprendizado evolutivo, evitando que um mesmo erro não seja cometido novamente. Porém, para atingir este aprendizado, é necessário que o conhecimento esteja disponível e acessível a toda organização de software. Neste contexto, sistemas de gerência de conhecimento podem ser bastante úteis. A gerência de conhecimento envolve recursos humanos, organização e cultura, além de tecnologia de informação, métodos e ferramentas para o seu apoio [2]. A gerência de conhecimento facilita a criação, acesso e reuso do conhecimento e seu objetivo principal é promover o surgimento de conhecimento novo, seu armazenamento e compartilhamento por toda a organização. Embora cada projeto de desenvolvimento de software seja único, experiências similares e comuns a eles podem ajudar os desenvolvedores a executarem suas atividades. O reuso deste conhecimento ajuda a evitar que falhas se repitam e auxilia na solução de problemas recorrentes. Mas, para ser efetiva a gerência de conhecimento deve ser integrada ao processo de software e suas várias tarefas.

Ontologias são bastante úteis para descrever e organizar conhecimento de domínio em um ADSOD. Porém, a construção de ontologias não é uma tarefa simples. Envolve a especificação de conceitos e relações que existem em um domínio de interesse, além de suas definições, propriedades e restrições, descritas na forma de axiomas [3]. Portanto, ferramentas para apoiar o desenvolvimento de ontologias são úteis em um ADSOD. Estas ferramentas devem suportar a definição e modificação de conceitos, propriedades, relações, axiomas e restrições, permitindo a inspeção, pesquisa e codificação das ontologias resultantes [4].

Este artigo discute como apoiar o desenvolvimento de software em um domínio específico, através da descrição do conhecimento do domínio usando ontologias e da provisão de serviços de gerência de conhecimento em ADSs, tendo por base o ambiente ODE (*Ontology-based software Development Engineering*) [5]. A seção 2 discute alguns aspectos de ADSODs e o uso de ontologias nestes ambientes. A seção 3 discute a importância da gerência de conhecimento e como ela pode ser utilizada em ADSs. A seção 4 apresenta o ambiente ODE e ODEd, um editor de ontologias, desenvolvido para apoiar a descrição e o compartilhamento de conhecimento em ODE. Na seção 5, é apresentado um estudo de caso de orientação ao domínio siderúrgico em ODE. Na seção 6, trabalhos correlatos são discutidos. Finalmente, a seção 7 apresenta as conclusões deste trabalho.

2 Ontologias e Ambientes de Desenvolvimento de Software Orientados a Domínio

O desenvolvimento de software é uma tarefa que envolve esforço coletivo, criativo e complexo. Assim, a qualidade do produto de software desenvolvido depende diretamente das pessoas, da organização e dos procedimentos utilizados para criar este produto. Em outras palavras, há uma direta correlação entre a qualidade do processo de software e a qualidade do produto desenvolvido [6]. Por esta razão, tanto pesquisadores quanto desenvolvedores têm dado bastante atenção ao entendimento e à melhoria do processo de software. Mas, para lidar com complexos processos de software, é necessário prover ferramentas computacionais para apoiar estas difíceis tarefas.

A necessidade de um apoio integrado para as atividades ao longo de todo o ciclo de vida do software fez surgir os Ambientes de Desenvolvimento de Software (ADSs). ADSs podem ser definidos como conjuntos de ferramentas integradas que facilitam as atividades da engenharia de software, apoiando todo o ciclo de vida do produto de software desenvolvido [7].

Contudo, uma das grandes dificuldades no desenvolvimento de software é que, muitas vezes, os desenvolvedores não estão familiarizados com o domínio para o qual o software está sendo desenvolvido. Para tratar este problema, vários grupos de pesquisa propuseram evoluir ADSs para apoiar o desenvolvimento de software considerando características peculiares do domínio [8,9,1,10]. A identificação da necessidade de apoiar o desenvolvimento de software orientado a domínio e as limitações dos ADSs convencionais em apoiar este tipo de desenvolvimento levou à definição de Ambientes de Desenvolvimento de Software Orientados a Domínio (ADSODs) [10].

Um ADSOD é um ADS que apóia o desenvolvimento de sistemas de software em um domínio específico, considerando seu conhecimento para guiar o desenvolvedor de software em várias tarefas do processo de software. Esta nova classe de ADS requer duas características essenciais: o conhecimento do domínio deve ser capturado,

modelado e armazenado para uso no ADS; e o ADS deve suportar a disseminação e uso do conhecimento do domínio.

Porém, é impossível representar o real mundo, ou até mesmo uma parte dele, em todos seus detalhes. Para representar um domínio é necessário focar em um número limitado de conceitos que são suficientes e pertinentes para se criar uma abstração dele. Assim, um aspecto central de qualquer atividade de modelagem consiste em desenvolver uma conceituação: um conjunto de regras informais que restringem a estrutura de um pedaço de realidade que um agente usa para isolar e organizar conceitos e relações pertinentes [11].

Para construir um ADSOD, é preciso definir um modelo que torne explícita a conceituação básica do domínio em questão. Ontologias têm sido usadas para este fim e, portanto, podem ser muito úteis para apoiar a orientação a domínio em um ADS.

De acordo com Guarino [12], uma ontologia é uma teoria lógica que responde pelo significado intencional de um vocabulário formal, i.e., seu compromisso ontológico com uma conceituação particular do mundo. Uma ontologia pode ter várias formas, mas, necessariamente, deve incluir um vocabulário de termos e alguma especificação do seu significado. Isto inclui definições e uma indicação de como conceitos estão inter-relacionados, o que coletivamente impõe uma estrutura no domínio e restringe as possíveis interpretações dos termos [13]. Assim, uma ontologia consiste de conceitos e relações, e suas definições, propriedades e restrições expressas na forma de axiomas [3].

Jasper et al. [14] classificaram as aplicações de ontologias em quatro categorias principais, enfatizando que uma aplicação pode integrar mais de uma destas categorias:

- Autoria neutra: uma ontologia é desenvolvida em uma única linguagem e é traduzida para diferentes formatos e usada em aplicações de múltiplos objetivos;
- Ontologias como Especificação: uma ontologia de um determinado domínio é criada e provê um vocabulário para especificar requisitos para uma ou mais aplicações-alvo. De fato, a ontologia é usada como uma base para especificação e desenvolvimento de software, permitindo reuso;
- Acesso Comum a Informação: uma ontologia é usada para permitir que múltiplas aplicações-alvo (ou pessoas) tenham acesso a fontes heterogêneas de informação que são expressas usando vocabulário diverso ou formato inacessível;
- Busca Baseada em Ontologias: uma ontologia é usada para procurar, em um repositório de informação, por recursos desejados, melhorando a precisão e reduzindo a quantidade de tempo gasto na busca.

No contexto de ADSODs, estamos interessados em todos esses usos potenciais de ontologias. O cenário de autoria neutra é importante, principalmente quando serão desenvolvidas aplicações usando tecnologias diferentes (por exemplo, objetos e lógica). Em um ADSOD, uma ontologia pode ser desenvolvida e, então, traduzida para componentes de implementação usando estas tecnologias. Assim, neste cenário, é necessário traduzir a ontologia para componentes de domínio.

A idéia básica do uso ontologias como especificação é conduzir ao reuso de software a níveis de abstração mais altos. Este cenário está claramente relacionado com as metas de um ADSOD. Se um ADSOD incorpora uma ontologia de domínio, aplicações neste domínio podem ser desenvolvidas com base em uma especificação compartilhada. Também, componentes de domínio, tais como frameworks e padrões de análise, que foram desenvolvidos com base em ontologias podem ser usados em outras atividades do processo de software.

Acesso comum à informação é essencial para melhorar a comunicação entre desenvolvedores e ferramentas em um ADS, evitando problemas de interpretação. Neste cenário, ontologias podem ser usadas para promover o entendimento comum entre desenvolvedores em um ADSOD. Uma ontologia é desenvolvida e os desenvolvedores de aplicação fazem um acordo ao usá-la. Para suportar este cenário, um editor de ontologias deve ser integrado ao ADSOD, apoiando a autoria e a navegação de ontologias. Finalmente, a busca baseada em ontologias tem grande potencial para melhorar estruturação e procura em bibliotecas de componentes disponíveis em um ADSOD.

Analisando estes cenários, podemos notar que ADSODs podem usufruir várias vantagens com o uso de ontologias. De fato, um ADSOD deve apoiar processos de software orientados a domínio. Alguns modelos de processo foram propostos com este propósito, quase sempre estabelecendo caminhos paralelos para a engenharia de domínio e a engenharia de software. Engenharia de domínio envolve estabelecer um conjunto de artefatos de software que podem ser usados para guiar o desenvolvimento de software no domínio correspondente. Seu propósito é identificar, modelar, construir, catalogar e disseminar um conjunto de artefatos que podem ser aplicados a qualquer software em um domínio de aplicação particular [15]. Na engenharia de domínio, ontologias podem agir como um modelo de domínio e um componente em um repositório de artefatos reutilizáveis, além de poderem ser usadas para estruturar este repositório.

Outro benefício do uso de ontologias no desenvolvimento de software é a reutilização de especificações de domínio na fase de especificação de requisitos. Em Engenharia de Software, para cada nova aplicação a ser construída, é

desenvolvida uma conceituação nova. Isto reflete em como requisitos são elicitados: para cada aplicação, uma fase de elicitação é realizada, quase sempre do nada, enfocando todas as particularidades do sistema. Esta abordagem é extremamente cara, já que a elicitação de requisitos é uma das atividades que consome mais tempo. Peritos são recursos escassos e caros e eles são essenciais a esta atividade. Portanto, deveriam ser mais bem utilizados. Por isso, é importante compartilhar e reutilizar o conhecimento capturado [16].

Em uma abordagem baseada em ontologias, a elicitação e a modelagem de requisitos podem ser realizadas em duas fases. Primeiro, o conhecimento de domínio geral deve ser extraído e especificado na forma de ontologias. Estas ontologias são usadas para guiar a segunda fase da análise de requisitos, quando são consideradas as particularidades de uma aplicação específica. Deste modo, a mesma ontologia pode ser usada para guiar o desenvolvimento de várias aplicações, diminuindo os custos da primeira fase e permitindo o compartilhamento e reuso do conhecimento [16].

Uma vez capturado o conhecimento na forma de ontologias, este deve ser disseminado e utilizado. A gerência de conhecimento pode prover um importante apoio a este processo. Mais além, não apenas o conhecimento do domínio, objeto de interesse direto de um ADSOD, mas também artefatos, discussões, decisões de projeto e lições aprendidas durante um projeto podem ser compartilhados por uma organização. Utilizando uma abordagem de gerência de conhecimento, o conhecimento criado durante o processo de software pode ser capturado, disseminado e reutilizado.

3 Gerência de Conhecimento em Ambientes de Desenvolvimento de Software

O conhecimento tem sido considerado como o patrimônio mais importante de uma organização e, portanto, deve ser cuidadosamente gerenciado. Historicamente, o conhecimento de uma organização tem sido registrado em papel ou guardado na cabeça das pessoas. Infelizmente, o conhecimento registrado em papel tem acessibilidade limitada e é difícil de ser atualizado. O conhecimento guardado na cabeça das pessoas, por sua vez, é perdido quando um indivíduo deixa a organização. Além disso, em uma grande corporação, pode ser difícil localizar a pessoa que conhece um determinado assunto. De fato, para a vantagem competitiva fornecida pelo conhecimento ser sustentável, o conhecimento não pode estar no nível de indivíduo [17]. Assim, o conhecimento tem de ser sistematicamente coletado, armazenado em uma memória corporativa e compartilhado por toda a organização.

A gerência de conhecimento trata de gerenciar formalmente os recursos de conhecimento, de modo a facilitar a criação, acesso e reuso de conhecimento, tipicamente usando tecnologia avançada. Ela é formal no sentido do conhecimento ser classificado segundo uma ontologia pré-especificada e armazenado em bases de dados [17].

As atividades da gerência de conhecimento incluem: identificação, captura, adaptação, integração, disseminação, uso e manutenção de conhecimento. No centro de um sistema de gerência de conhecimento, está uma memória organizacional, apoiando o compartilhamento de conhecimento organizacional, incluindo lições aprendidas [18].

Ontologias são particularmente importantes para a gerência de conhecimento. São precisamente elas que estabelecem uma ligação estreita entre as atividades da gerência de conhecimento, permitindo uma visão orientada a conteúdo [19]. Ontologias definem um vocabulário compartilhado a ser utilizado no sistema de gerência de conhecimento para facilitar a comunicação, a integração, à busca e o armazenamento do conhecimento [17].

De acordo com Borges et al. [20], o conhecimento pode ser classificado como formal ou informal. O conhecimento formal pode ser expresso de uma forma estruturada e é fácil de ser comunicado e compartilhado. Como exemplo de conhecimento formal podem ser citados os artefatos de software, componentes, padrões, entre outros. O conhecimento informal é altamente pessoal e difícil de se formalizar, tornando difícil o seu compartilhamento. O conhecimento informal surge a partir das experiências individuais e envolve fatores como crença pessoal, perspectivas e valores. Exemplos deste tipo de conhecimento são as discussões e as lições aprendidas.

No contexto do desenvolvimento de software, as lições aprendidas podem ser consideradas um dos principais tipos de conhecimento. As lições são criadas como resultado do trabalho da própria organização. Devem descrever tanto os relatos de sucesso quanto oportunidades de melhoria. Lições de sucesso relatam boas respostas a situações ocorridas. Oportunidades de melhoria relatam o que ocorreu de errado em uma situação e formas potenciais para solucionar o problema [17]. O reuso de lições aprendidas em projetos passados promove boas práticas de desenvolvimento de software e previne a repetição de erros.

Uma gerência de conhecimento eficiente deve ser capaz de apoiar a criação, captura e utilização dos vários tipos de conhecimento. Os objetivos de uma organização determinam o tipo de conhecimento que ela deve capturar. Qualquer tipo de conhecimento pode ser utilizado para evitar re-trabalho e melhorar a qualidade. Processos, modelos de qualidade, artefatos desenvolvidos, experiências e lições aprendidas são alguns exemplos de conhecimento reutilizáveis. Porém, para que o reuso de conhecimento seja eficiente em uma organização, é necessário um armazenamento adequado deste conhecimento. Por exemplo, um item de conhecimento gerado em um projeto deve ser adaptado a futuras necessidades de futuros projetos, agregando informações que auxiliem seu reuso [21]. Estes objetivos podem ser alcançados através de um sistema de gerência de conhecimento eficiente.

Para ser efetivamente utilizado, um sistema de gerência de conhecimento tem de estar integrado ao processo da organização, permitindo que o conhecimento relevante seja coletado e armazenado à medida que ele é gerado no trabalho. Conseqüentemente, o sistema de gerência de conhecimento deve estar integrado ao ambiente de trabalho existente [18]. No contexto de desenvolvimento de software, esse ambiente de trabalho é exatamente um Ambiente de Desenvolvimento de Software e, portanto, ADSs e gerência de conhecimento complementam um ao outro no apoio ao desenvolvedor durante o processo de software para a criação de produtos de melhor qualidade e com maior produtividade.

4 ODE: Um Ambiente de Desenvolvimento de Software Baseado em Ontologias

ODE (*Ontology-based software Development Environment*) [5] é um Ambiente de Desenvolvimento Centrado em Processo, desenvolvido no Laboratório de Engenharia de Software da Universidade Federal do Espírito Santo (LabES/UFES), tendo por base ontologias. ODE busca combinar técnicas, métodos e ferramentas para apoiar o desenvolvedor na construção de produtos de software, abrangendo todas as atividades inerentes ao processo, tais como planejamento, gerência, desenvolvimento e controle da qualidade.

A premissa do projeto ODE está baseada no seguinte argumento: se as ferramentas em um ADS são construídas baseadas em ontologias, a integração de ferramenta pode ser mais facilmente obtida. A mesma ontologia pode ser usada para construir ferramentas diferentes que apóiam atividades de engenharia de software correlacionadas. Além disso, se as ontologias são integradas, a integração de ferramentas construídas com base nelas pode ser bastante facilitada. Esta abordagem foi usada para tratar a integração de processo e a integração de conhecimento de qualidade de software [22].

Porém, o problema da integração ainda está sendo objeto de estudo. A integração do conhecimento de domínio deve ser considerada para prover apoio de gerência de conhecimento e evoluir ODE para um ADSOD. Sendo assim, foi identificada a necessidade de prover suporte ao desenvolvimento de ontologias de domínio em ODE, como forma de permitir a descrição do conhecimento de domínio no ambiente. Além disso, uma vez que a meta é promover o reuso em diversos níveis de abstração, é importante que o ambiente tenha um conjunto de componentes de domínio que possa ser utilizado em diferentes etapas do desenvolvimento.

Já que, atualmente, o paradigma principal da Engenharia de Software é a tecnologia de objetos, vale a pena codificar as ontologias resultantes utilizando objetos, como forma de prover uma biblioteca de componentes de domínio. Falbo et al. [23] propuseram uma abordagem ontológica para a engenharia domínio, tomando por base a proposta de Arango e Prieto-Díaz [24], que considera três atividades principais: análise de domínio, especificação de uma infra-estrutura de reuso e implementação dessa infra-estrutura. Essa abordagem ontológica envolve o desenvolvimento de ontologias (análise de domínio), seu mapeamento para modelos de objetos (especificação de infra-estrutura) e o desenvolvimento de componentes Java (implementação de infra-estrutura). A primeira fase - desenvolvimento de ontologias - envolve as seguintes atividades [3,23]: identificação de propósito e especificação de requisitos, captura e formalização da ontologia, integração de ontologias existente, avaliação da ontologia e documentação.

Para derivar *frameworks* de objetos a partir de ontologias (especificação de infra-estrutura e implementação), a abordagem define um conjunto de diretivas, padrões de projeto e regras de transformação [16]. As diretivas são usadas para guiar o mapeamento da estrutura epistemológica da ontologia de domínio (conceitos, relações, propriedades e papéis) para suas contrapartes no paradigma orientado a objetos (classes, associações, atributos e papéis). Padrões de projeto e regras de transformação são aplicados no mapeamento de axiomas. A aplicação destas diretrizes é apoiada por um *framework* Java chamado "Set", que implementa o tipo matemático conjunto. A linguagem Java foi escolhida por implementar todos os conceitos de orientação a objetos necessários à cobertura dos requisitos de implementação das estruturas do modelo e por ser independente de plataforma. [16].

Para apoiar o processo de engenharia de domínio baseado em ontologias descrito acima em ODE, foi construído ODEd (*ODE's Ontology Editor*). A meta principal de ODEd é apoiar desenvolvimento de ontologias e assim permitir compartilhamento de conhecimento do domínio em ODE. ODEd apóia a análise de domínio, suportando a definição de conceitos e relações como uso de representações gráficas, e promovendo a geração automática de alguns tipos de axiomas. ODEd suporta projeto de domínio derivando *frameworks* de objetos a partir das ontologias.

5 Orientação ao Domínio Siderúrgico em ODE

A falta de conhecimento sobre o domínio com o qual se está trabalhando é um problema que atinge grande parte das organizações. No caso das companhias siderúrgicas, este problema é especialmente grave. Muitas das atividades do processo de software são prejudicadas pela particularidade do assunto, o domínio siderúrgico, pela rotatividade de funcionários, que constantemente trocam de departamento ou de funções dentro de uma mesma área, pela rotatividade de empresas prestadoras de serviços e a crescente ampliação do domínio em questão, com agregação de

novas etapas ao processo produtivo. Novos profissionais admitidos ou contratados, geralmente, não têm conhecimento sobre o domínio, sendo necessário que os profissionais mais experientes façam uma apresentação do seu conhecimento sobre o domínio e sobre os sistemas atuais, desde o aceite de pedidos até o despacho dos produtos produzidos. Estes profissionais mais experientes, normalmente, estão envolvidos em outros projetos e atividades, o que faz com que a transferência do conhecimento seja lenta [25].

Tendo em vista que ADSODs visam minimizar os problemas acima apontados, a construção de um ADSOD para o domínio siderúrgico pode ser benéfica para a organização, pois permite que o conhecimento do domínio siderúrgico seja incorporado ao ADSOD, facilitando o aprendizado daqueles que não o conhecem [25]. Um ADSOD para o domínio siderúrgico tem a finalidade de auxiliar o aprendizado do domínio, tanto para a equipe que iniciou o projeto quanto para os que chegam à equipe durante o desenvolvimento.

Conforme discutido anteriormente, em ODE, a orientação a domínio tem por base a construção de uma ontologia de domínio, a derivação de um framework a partir da mesma e a provisão de serviços de gerência de conhecimento para facilitar o reuso e o compartilhamento do conhecimento do domínio. Para apoiar a realização dessas tarefas, está disponível o editor de ontologias ODEd. Para exemplificar o desenvolvimento de ontologias e a geração de *frameworks* em ODEd, a seguir é apresentada construção de uma ontologia de processo siderúrgico na ferramenta.

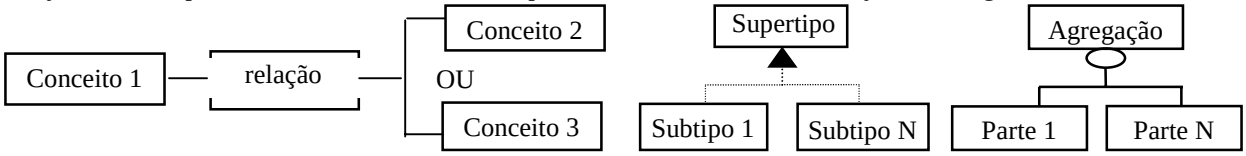
5.1 Uma Ontologia do Domínio Siderúrgico

O objetivo da construção de uma ontologia siderúrgica é prover um entendimento compartilhado do conhecimento, para ela apóie a aquisição, organização, reuso e compartilhamento de conhecimento sobre o domínio em questão. A intenção é que, através da utilização da ontologia, a construção de novos sistemas e a manutenção dos sistemas atuais possam contar com maior facilidade de comunicação entre os desenvolvedores, facilidade de reutilização, maior confiabilidade em relação aos conceitos e relações e maior inter-operabilidade entre sistemas [25].

O primeiro passo do desenvolvimento de uma ontologia é a identificação do propósito e a especificação de sua competência. Para apoiar esta fase, ODEd permite que o usuário defina questões de competência. Dentre as questões que a ontologia de processo siderúrgico deve responder estão:

- Que produtos são gerados por um processo?
- Que produtos ou matérias primas podem ser insumos para um novo processo?
- Como as matérias primas são classificadas?
- Como os produtos são classificados?

Uma vez definidas as questões de competência, é possível começar a etapa de captura. Nesta etapa, o uso de uma representação gráfica é essencial para facilitar a comunicação entre os engenheiros de software e os especialistas de domínio. Falbo et al. [3] propuseram LINGO - uma LINguagem Gráfica para representar Ontologias. LINGO possui primitivas básicas para representar uma conceituação de domínio, i.e., em sua forma mais simples, suas notações representam somente conceitos e relações. Qualquer notação de LINGO, além das notações básicas para conceitos e relações visa incorporar uma teoria [3]. Deste modo, podem ser gerados axiomas automaticamente. ODEd usa esta característica para gerar estes tipos de axiomas automaticamente. Em deste modo, ODEd embute um poderoso mecanismo de inclusão de teorias. Cada tipo de relação especificando uma teoria genérica tem sua própria notação e, sempre que é usada, ontologias genéricas são automaticamente integradas [3]. A Figura 1 mostra as principais notações de LINGO e alguns dos axiomas impostos pela relação todo-parte. Axiomas de anti-reflevividade (A1), anti-simetria (A3) e transitividade (A4) denotam propriedades suficientes e necessárias para todos os tipos de relações de todo-parte. Os axiomas restantes completam a teoria definindo distinções ontológicas satisfatórias.



(A1) $\forall x \ \neg parteDe(x,x)$
(A2) $\forall x,y \ parteDe(y,x) \leftrightarrow todoDe(x,y)$
(A3) $\forall x,y \ parteDe(y,x) \rightarrow \neg \ parteDe(x,y)$
(A4) $\forall x,y,z \ parteDe(z,y) \wedge parteDe(y,x) \rightarrow parteDe(z,x)$
(A5) $\forall x,y \ disjunto(x,y) \rightarrow \neg\exists z \ parteDe(z,x) \wedge parteDe(z,y)$
(A6) $\forall x \ atomico(x) \rightarrow \neg\exists y \ parteDe(y,x)$

Figura 1. Principais notações de LINGO.

Como o Domínio Siderúrgico é bastante amplo e complexo, uma sub-divisão do mesmo se faz necessária, para facilitar a compreensão e assegurar que a ontologia não se desvie da sua finalidade inicial. A fim de atingir esse objetivo, é necessário garantir o critério de comprometimento ontológico mínimo, ou seja, tornar a ontologia mais genérica possível, contendo apenas elementos essenciais e evitando a agregação de detalhes que tornarão a reutilização mais difícil [25].

Dessa forma, identificou-se a necessidade de, antes de se construir uma ontologia de processo siderúrgico, construir uma *Ontologia de Processo Produtivo* [25], tratando de maneira mais geral os aspectos relativos a um processo produtivo geral, o que envolve insumos, equipamentos e produtos, dentre outros. Sendo assim, foi utilizada a abordagem de ontologia em níveis [3], ou seja, a ontologia de processo siderúrgico foi construída tendo por base a ontologia de processo produtivo.

ODEd suporta a integração de ontologias de um modo muito simples. É possível importar conceitos de ontologias existentes para a ontologia corrente. Se mais de um conceito é importado e há relações entre eles, estas relações também são incorporadas à ontologia. Então, os conceitos importados podem ser conectados aos conceitos da ontologia corrente.

Na Figura 2, por exemplo, os conceitos *equipamento*, *processo*, *matéria-prima*, *produto* e sua hierarquia foram importados da ontologia de processo produtivo para a ontologia de processo siderúrgico. Já que a relação *insumo* se estabelece entre conceitos importados (*processo*, *matéria-prima* e *produto*), ela também foi incorporada à ontologia. Depois que o conceito *matéria-prima* foi incorporado, uma hierarquia de matérias-primas foi cria para tratar especificamente do domínio siderúrgico. Assim, os conceitos *Minério de Ferro, Carvão, Fundentes, Sucata de Aço e Ferro-ligas* foram criados na ontologia de processo siderúrgico como sub-tipos do conceito *Matéria-prima* da ontologia de processo produtivo.

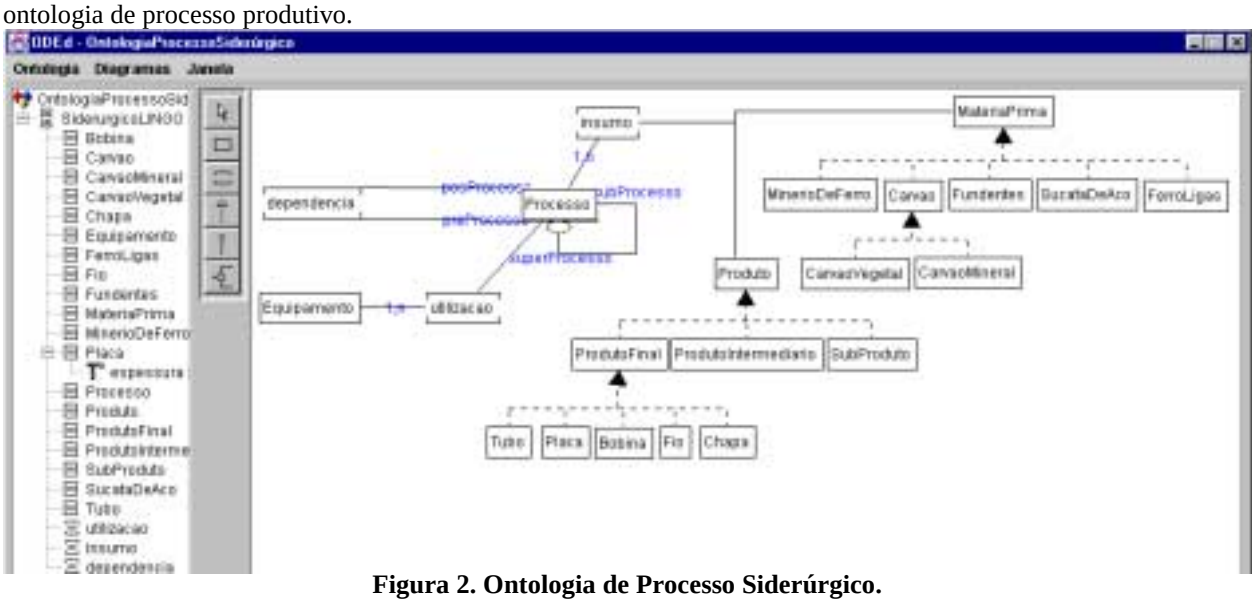


Figura 2. Ontologia de Processo Siderúrgico.

A Figura 2 mostra apenas parte da *Ontologia de Processo Siderúrgico* [25], contemplando os aspectos descritos a seguir. Em um *processo* produtivo genérico, um determinado conjunto de insumos é processado através dos *equipamentos*, obedecendo à funcionalidade do processo, gerando produtos. Processos são decompostos em outros processos menores. Um *sub-processo* é um processo que compõe um processo maior e um *super-processo* é um processo que é composto por processos menores. No domínio siderúrgico, um exemplo de super-processo é o processo de redução, que é composto pelos processos de sinterização e coqueria. Processos devem ser realizados segundo uma ordem específica, ou seja, não podem ser realizados aleatoriamente. Assim, é importante compreender como os processos são encadeados. Para tal foram introduzidos os papéis de *pré-processo* e *pós-processo*.

Dependendo do propósito do processo no qual são gerados, os *produtos* podem ser classificados em: produtos intermediários, produtos finais e sub-produtos. Um *produto intermediário* é aquele obtido num determinado processo e que não será comercializado. Ele servirá de insumo para um outro processo. No processo de redução, por exemplo, o produto obtido é o ferro gusa. Tal produto posteriormente é utilizado no processo de refino para produção do aço. Dessa forma, o ferro gusa é um produto intermediário. *Produto final* é aquele que é o objetivo do processo produtivo com um todo. Numa empresa que comercializa bobinas de aço, o produto final é a própria bobina. *Sub-produto* é aquele produto gerado por um determinado processo e que não tem nenhuma utilização direta. Ele pode até ser comercializado, mas sua geração não é um objetivo da empresa. Durante o processo de redução, por exemplo, é gerada a escória de alto forno. Tal produto não tem utilização dentro do processo siderúrgico, o que não impede que ele seja utilizado pela indústria de construção civil.

Para que um determinado processo possa ocorrer, é necessário que ele seja alimentado com determinados *insumos*. Tais insumos podem ser matérias primas *ou* produtos produzidos em processos anteriores. Um processo pode ter vários insumos, como ocorre no processo de redução que possui como insumos minério de ferro e carvão mineral. As *matérias primas* são os materiais utilizados como insumo básico para um processo produtivo em sua forma natural. No domínio siderúrgico, as matérias primas podem ser classificadas em: *minério de ferro, carvão,*

fundentes, sucata de aço e ferro-ligas, sendo que os carvões se dividem, ainda, em: *carvão mineral e carvão vegetal*. Os produtos finais possuem várias classificações. Neste trabalho foi adotada a seguinte classificação: *tubo, placa, bobina, fio e chapa*.

Cardinalidades são utilizadas no diagrama para mostrar quantas instâncias de um conceito podem participar na relação. Na figura 2, a cardinalidade (1,n) na relação *utilização* implica que um equipamento sempre tem que ser utilizado por, pelo menos, um processo: $(\forall e)\ (equipamento(e) \rightarrow (\exists p)\ (processo(p) \wedge utilizacao(e,p))$. Já que a cardinalidade (0,n) não impõe restrições, ela não é representada. Alguns conceitos e relações possuem propriedades. Na Figura 2, *placa* tem a propriedade *espessura*, mostrada no lado esquerdo da figura.

Além de gerar teorias pré-definidas nas notações da linguagem, ODEd também permite ao usuário compor teorias e aplicá-las a relações da ontologia. Nesta abordagem, axiomas são classificados de acordo com certas propriedades de axioma, a saber: *anti-reflexividade, anti-simetria, atomicidade, disjunção, exclusividade, reflexividade, simetria e transitividade*. Estas propriedades são usadas para compor teorias associadas a relações na ontologia.

É possível, por exemplo, associar uma teoria à relação de *dependência* na ontologia siderúrgica. Esta relação tem as seguintes propriedades: transitividade- se *p1*é um pré-processo de *p2* e *p2* é um pré-processo de *p3*, então *p1*é pré-processo de *p3*; anti-simetria - se *p1*é um pré-processo de *p2*, então *p2* não pode ser pré-processo de *p1*; e anti-reflexividade - *p1* não pode ser pré-processo de si mesmo. ODEd permite a representação de axiomas a partir dessas teorias, tais como discutido acima e mostrado na Figura 3.

Figura 3. Teoria da relação Dependencia.

5.2 O Framework da Ontologia de Processo Siderúrgico

No desenvolvimento para reuso, é importante observar o conceito de produto reutilizável de forma mais abrangente. A reutilização deve acontecer através de artefatos situados em vários níveis de abstração. Dessa forma, a construção de um *framework* mostra-se bastante útil, pois provê itens que podem ser reutilizados por um processo de engenharia de software para problemas inseridos em um domínio específico, diminuindo o trabalho de projeto e implementação [23]. Um *framework* pode ser considerado como um projeto reutilizável de alto nível, consistindo de classes que são especialmente projetadas para serem refinadas e usadas em grupo [26]. Assim, no contexto desse trabalho, um *framework* deve ser entendido como uma estrutura computacional passível de extensão para aplicações dentro de um domínio específico. A reutilização de infra-estruturas de domínio possui várias vantagens, sendo as principais delas ligadas à comunicação e ao entendimento do domínio em questão e à obtenção de uma estrutura estável, confiável, robusta e com alto grau de manutenibilidade. No entanto, a obtenção desses benefícios depende da qualidade da infra-estrutura utilizada, que, por sua vez, depende diretamente de seu modelo de domínio [25].

O modelo de domínio para o domínio siderúrgico constitui-se da ontologia apresentada anteriormente. Contudo, para facilitar o reuso, a ontologia produzida deve dar origem a uma infra-estrutura computacional, passível de posterior reutilização por um processo convencional de engenharia de software. Assim, a partir da ontologia de processo siderúrgico, foi derivado um *framework*, mostrado na Figura 4, utilizando o paradigma da orientação a objetos. Este framework foi gerado em ODEd, que suporta o mapeamento de ontologias para um modelo de objetos, seguindo a abordagem proposta em [16]. Conceitos e relações são naturalmente mapeados, respectivamente, em classes e associações em um modelo de objeto. Relações entre três ou mais conceitos e relações com propriedades dão origem a classes associativas [16]. No caso da ontologia siderúrgica, foram derivadas, por exemplo, as classes `Processo` e `Equipamento` a partir dos conceitos correspondentes, como também as associações `utilizacao` e `dependencia`, como mostrado na Figura 4. Auto-relacionamentos, como `dependencia`, também são mapeados como associações e geram um método para cada fim de associação. O nome destes métodos é, em vez do nome da relação, o nome dos papéis exercidos pelo conceito. Assim, a relação *dependência* origina os métodos

`preProcesso()` e `posProcesso()`.

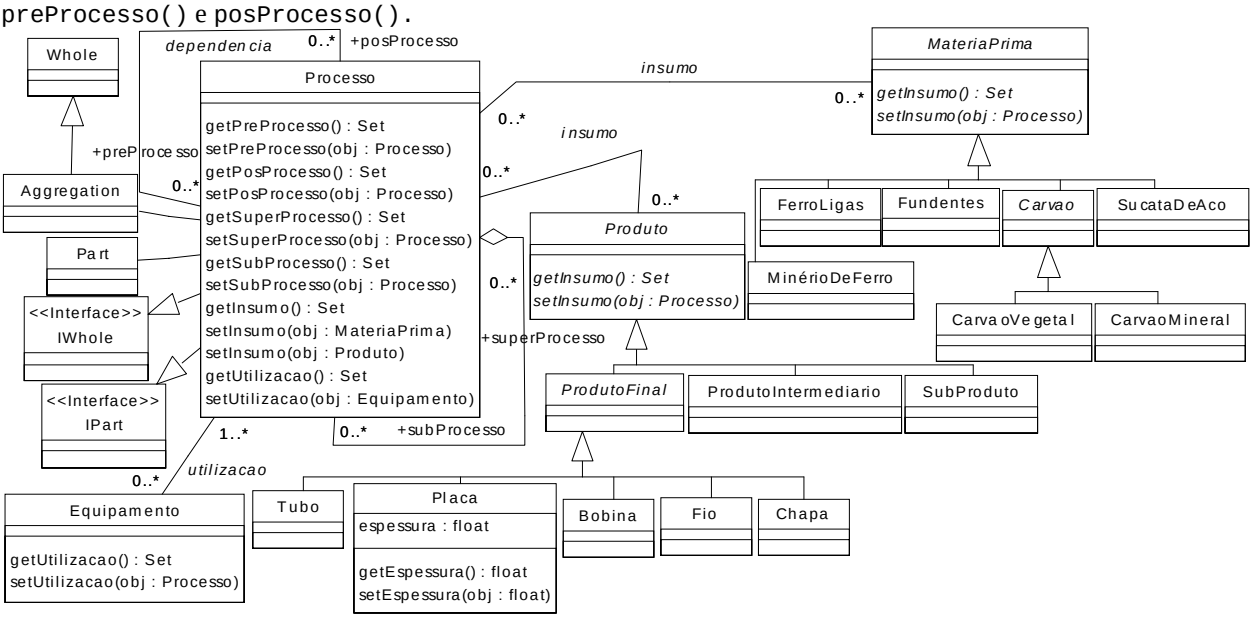


Figura 4. O Framework da Ontologia de Processo Siderúrgico gerado por ODEd.

Propriedades de conceitos e relações são mapeadas como atributos das classes correspondentes [16]. A propriedade *espessura* do conceito *Placa*, por exemplo, foi mapeada como o atributo `espessura` na classe `Placa`. Além disso, para cada atributo, foram criados métodos para retornar e inserir seus valores (`getEspessura` e `setEspessura`).

No nível ontológico, o direcionamento das relações é abstraído e, por este motivo, para manter o mesmo grau de generalidade, nos diagramas de classes, os relacionamentos são implementados como bidirecionais, fazendo com que, em ambas as classes, exista um método com o nome da relação [16]. Assim, a relação *utilização* é implementada como um atributo do tipo `Set` nas classes `Equipamento` e `Processo`, e há métodos `utilizacao` em ambas as classes. As hierarquias de tipos (taxonomias) e seus correspondentes axiomas não requerem nenhuma implementação adicional e são representados diretamente em relações de especialização/generalização [16].

A notação UML para agregação não garante as restrições impostas pela relação todo-parte. Para lidar com este problema, Guizzardi et al. [16] propuseram o Padrão *Whole-Part*. Neste padrão, a classe `Whole` pode garantir à suas classes concretas associadas a verificação de um conjunto satisfatório de restrições antes que uma relação entre elas seja estabelecida. As interfaces que `IWhole` e `IPart` devem ser implementados pelas classes concretas. No *framework* derivado da ontologia de processo siderúrgico (Figura 4), a classe `Processo` está relacionada a `IWhole` e `IPart`, já que sua relação todo-parte é reflexiva. Dá mesma forma, `Processo` está relacionado aos manipuladores `Aggregation` e `Part`. Antes que um processo seja adicionado como sub-processo de outro, é preciso verificar se as restrições da relação são mantidas. Esta checagem é feita por estes manipuladores.

Padrões de projeto também são utilizados para representar alguns tipos de axiomas, como transitividade e anti-simetria. Estes padrões são capturados através de classes capazes conferir se as propriedades do axioma são verdadeiras. Assim, a classe `Processo` está relacionada aos padrões `AntiSymmetry`, `AntiReflexivity` e `Transitivity`. Antes de fixar um processo como pré-processo de outro, a teoria da relação *dependência*, composta por estes axiomas, é checada através destes padrões.

5.3 Navegando as Ontologias

Com a construção da ontologia de processo siderúrgico, é possível prover um entendimento compartilhado a respeito do domínio, melhorando a comunicação entre os desenvolvedores e usuários, uma vez que as ambigüidades são minimizadas. Para promover o entendimento comum entre desenvolvedores em ODE, ODEd provê geração a automática de tutoriais de domínio baseados nas ontologias projetadas. Usando estes tutoriais, o desenvolvedor pode pesquisar e procurar os conceitos, relações e restrições do domínio.

A linguagem escolhida para construir os documentos foi XML [27] porque ela permite definir a sintaxe de documentos estruturados. Além disso, o esquema XML e ontologias têm uma meta comum: prover um vocabulário e estrutura para descrever a informação ser trocada. Os tutoriais são apresentados ao usuário como documentos HTML. Para isso, o editor usa XSL (*eXtensible Style sheet Language*), um documento de transformação e linguagem de formatação [28].

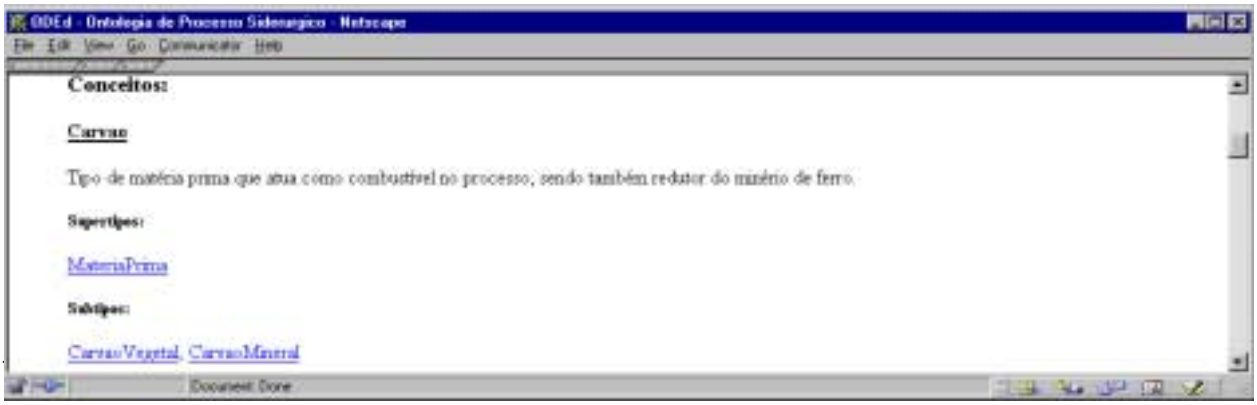


Figura 5. Navegando na Ontologia de Processo Siderúrgico.

A Figura 5 mostra o tutorial derivado a partir da ontologia de processo siderúrgico. É possível visualizar os conceitos e relações da ontologia e suas definições e propriedades. Do conceito *Carvão*, por exemplo, o usuário pode pesquisar seus super e sub-conceitos e visualizar suas definições.

5.4 Uma Infra-Estrutura para Gerência de Conhecimento

Para apoiar as atividades que compõem a gerência de conhecimento, uma infra-estrutura deve ser provida. A memória corporativa ou organizacional deve estar localizada no centro desta infra-estrutura, apoiando o armazenamento e compartilhamento do conhecimento. Posicionados ao redor da memória organizacional, os serviços de gerência devem prover ativamente informações úteis a usuários que estejam trabalhando em atividade que exijam conhecimento [18].

O primeiro requisito para o desenvolvimento de uma memória organizacional é promover a melhoria da acessibilidade ao conhecimento da organização, através de um repositório centralizado e bem estruturado. Sob este ponto de vista, as ontologias desenvolvidas em ODEd possuem um papel de extrema importância. As ontologias são utilizadas para estruturar e organizar a memória organizacional, ou seja, é possível relacionar itens de conhecimento com conceitos ontológicos. Além disso, na abordagem de gerência de conhecimento de ODE, instâncias de uma ontologia também desempenham o papel de itens de conhecimento, já que uma ontologia pode ser instanciada através do editor ODEd. Assim, esses itens também podem ser reutilizados.

Uma vez que os membros da organização estão freqüentemente muito ocupados para procurar uma informação, ou ainda, desconhecem que uma informação relevante existe, devem existir serviços pró-ativos que lembrem a estes usuários a existência de conhecimento útil. Desta forma, a distribuição do conhecimento pode ser ativa ou passiva, pois o usuário pode tomar a iniciativa de buscar uma informação ou o próprio sistema de gerência de conhecimento pode oferecer o conhecimento que julgar relevante à tarefa que o usuário está executando. Neste contexto, agentes podem ser utilizados e, novamente, as ontologias desenvolvidas são bastante úteis, uma vez que definem um vocabulário comum para a comunicação entre agentes.

Para obter a aceitação do usuário, o sistema de gerência de conhecimento deve estar integrado ao processo da organização, permitindo a coleta e o armazenamento de itens de conhecimento, assim que eles são gerados durante o trabalho. A abordagem de ODE, na qual a gerência de conhecimento é uma facilidade integrada ao ambiente, favorece esta integração ao processo de trabalho. Além disso, para manter a memória organizacional atualizada, é importante obter feedback de seus usuários, que são capazes de apontar deficiências e sugerir melhorias, integrando cada vez mais o sistema de gerência de conhecimento ao seu trabalho diário. Baseando-se no feedback do usuário, é possível, então, não apenas a manutenção da memória organizacional, mas também a sua evolução.

6 Trabalhos Correlatos

Alguns trabalhos têm procurado oferecer apoio ao desenvolvimento em domínios específicos através de ADSODs, tais como [1,9,10,25], e gerência de conhecimento, como em [9,28]. No contexto da *Estação TABA*, um meta-ambiente capaz de gerar, através de instanciação, ambientes de desenvolvimento de software adequados às necessidades de processos de desenvolvimento e de projetos específicos, o apoio à orientação a domínio tem sido bastante estudado [10]. ADSODs instanciados para um mesmo domínio podem utilizar as mesmas ferramentas construídas especificamente para este domínio, podendo utilizar também os componentes de software desenvolvidos. A base para a orientação a domínio no TABA é a definição de Teorias de Domínio [1]. Para permitir a incorporação de uma Teoria do Domínio, foi definida e implementada a ferramenta *EDITED* [1], que permite a entrada de informações de uma ontologia e gera classes para cada conceito da mesma. As relações de sub-tipo entre conceitos são geradas automaticamente por EDITED, porém as classes não embutem as restrições dos axiomas. Todos os axiomas devem ser codificados pelo usuário em Prolog e informados. ODEd trabalha de forma similar: são

geradas classes a partir dos conceitos e relações da ontologia. Contudo, já que ODEd adota LINGO como linguagem gráfica, os axiomas embutidos nas notações gráficas e aqueles axiomas de teorias definidas pelo usuário são incorporados automaticamente ao *framework*.

Ainda no contexto da Estação TABA, foi desenvolvido *SIDER* [25], um ADSOD para o domínio siderúrgico, utilizando a ontologia de processo siderúrgico, apresentada anteriormente. SIDER é um ambiente instanciado TABA, automaticamente gerado a partir da definição da Teoria de Domínio e de um processo de software. ODE, por sua vez, não é um ambiente construído especificamente para o domínio siderúrgico como SIDER, mas um ADS configurável, permitindo que se trabalhe em projetos de diferentes domínios no mesmo ambiente.

Em [9], são apresentados ADSODs com gerência de conhecimento integrada. *NetDE*, por exemplo, possui um sistema de gerência de conhecimento que apóia a criação e a administração de conhecimento no domínio de projeto e administração de rede de área local. NetDE apóia acesso de informação por busca, mas pode ter um papel ativo na disseminação de conhecimento. Em ODE, uma infra-estrutura para gerência de conhecimento está sendo construída para permitir o desenvolvimento de sistemas de gerência de configuração integrados ao ambiente.

Ontologias têm sido apontadas como sendo cruciais para ADSODs [1] e sistemas de gerência de conhecimento. Benjamins et al. [29], por exemplo, apresentam uma abordagem para gerência de conhecimento baseada em ontologias e utilizam ontologias para a organização e estruturação do conhecimento. Staab et al. [19] apresentam uma abordagem de gerência de conhecimento baseada em ontologias que distingue o processo de conhecimento (que trata de itens de conhecimento) do meta-processo de conhecimento (que introduz e mantém os sistemas de gerência de conhecimento. Esta abordagem também é utilizada em ODE. Ontologias são utilizadas para organizar o conhecimento e as ferramentas integradas ao ambiente são construídas baseadas em ontologias.

7 Conclusões

O desenvolvimento de software não é uma tarefa simples e requer apoio automatizado. Neste artigo, foram discutidos a importância dos ambientes de desenvolvimento de software e os benefícios em se integrar gerência de conhecimento em um ADS para prover um apoio ainda mais efetivo. Foi apresentado ODE, um ambiente de desenvolvimento centrado em processo e baseado em ontologias. Um de seus objetivos principais é promover o reuso de componentes de domínio, em diferentes etapas do desenvolvimento.

Com um sistema de gerência de conhecimento integrado ao processo de software, qualquer tipo de conhecimento pode ser reutilizado com intuito de evitar retrabalho e melhorar a qualidade do produto desenvolvido. A abordagem de gerência de conhecimento utilizada baseia-se fortemente em ontologias, desenvolvidas em ODEd, seja para organizar sua memória organizacional, seja para definir os itens de conhecimento. ODEd é um editor de ontologias que apóia desenvolvimento de ontologia usando representações gráficas, além de promover a geração automática de alguns tipos de axiomas e a derivação de *frameworks* a partir das ontologias. Utilizando ODEd para desenvolver ontologias e derivar *frameworks* de objetos a partir delas, está-se facilitando não só reuso de componentes de software, mas também o reuso de conhecimento em um ADS.

Agradecimentos

Os autores agradecem a CAPES pelo apoio financeiro a este trabalho.

Referências

[1] Oliveira, K.M., Modelo para Construção de Ambientes de Desenvolvimento de Software Orientados a Domínio, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, Brasil, 1999.

[2] O'Leary, D.E., Studer, R., Knowledge Management: An Interdisciplinary Approach, *IEEE Intelligent Systems*, Vol. 16, No. 1, Janeiro/Fevereiro de 2001.

[3] Falbo, R.A., Menezes, C.S., Rocha, A.R.C., A Systematic Approach for Building Ontologies, *Proceedings of the IBERAMIA'98*, Lisboa, Portugal, 1998.

[4] Lassila, O., Van Harmelen, F., Horrocksm, I., Hendler, J., Mcguinness, D. L., The Semantic Web and its Languages, *IEEE Intelligent Systems*, Novembro/Dezembro de 2000.

[5] Falbo, R.A, Guizzardi, G., Natali, A.C.C., Bertollo, G., Ruy, F.B., Mian, P.G.,Towards Semantic Software Engineering Environments, *Proc. of 14th Int. Conference on Software Engineering and Knowledge Engineering*, SEKE'02, Ischia, Itália, 2002.

[6] Fuggetta, A., Software Process: A Roadmap, *Proc. of The Future of Software Engineering*, ICSE'2000, Limerick, Irlanda, 2000.

[7] Harrison, W., Ossher, H., Tarr, P., Software Engineering Tools and Environments: A Roadmap, *Proc. of The Future of Software Engineering*, ICSE'2000, Irlanda, 2000.

[8] Fischer, G., Domain-Oriented Design Environments, *The Int. Journal of Automated Reasoning and Artificial Intelligence in Software Engineering*, Vol. 1, No 2, Junho de 1994.

[9] Fischer, G., Ostwald, J., Knowledge Management: Problems, Promises, Realities, and Challenges, *IEEE Intelligent Systems*, Janeiro/Fevereiro de 2001.

[10] Oliveira, K.M., Travassos, G.H., Menezes, C.S., Rocha, A.R.C., Ambientes de Desenvolvimento de Software Orientados a Domínio, *Proceeding of the XIV Simpósio Brasileiro de Engenharia de Software*, 2000.

[11] Guarino, N., Understanding, Building, and Using Ontologies, *International Journal of Human and Computer Studies*, 46, p. 293-310, 1997.

[12] Guarino, N., Formal Ontology and Information Systems, N. Guarino (Ed.), *Formal Ontologies in Information Systems*, IOS Press, 1998.

[13] Uschold, M., Knowledge level modelling: concepts and terminology, *Knowledge Engineering Review*, vol. 13, no. 1, 1998.

[14] Jasper, R., Uschold, M., A Framework for Understanding and Classifying Ontology Applications, *Proc. of the 12th Workshop on Knowledge Acquisition*, Modeling and Management (KAW'99), Alberta, Canada, 1999.

[15] Pressman, R.S., *Software Engineering: A Practitioner's Approach*, Fischer, 5th edition, New York: McGraw-Hill, 2000.

[16] Guizzardi, G., Falbo, R.A., Pereira Filho, J.G., Using Objects and Patterns to Implement Domain Ontologies, *Anais do XV Simpósio Brasileiro de Engenharia de Software*, Outubro de 2001.

[17] O'Leary, D.E., Enterprise Knowledge Management, *IEEE Computer Magazine*, March 1998.

[18] Abecker, A., Bernardi, A., Hinkelman, K., Toward a Technology for Organizational Memories, *IEEE Intelligent Systems*, Vol. 13., No. 3, pp. 40-48, Maio/Junho de 1998.

[19] Staab, S., Studer, R., Schurr, H. P., Sure, Y., Knowledge Processes and Ontologies, *IEEE Intelligent Systems*, Vol. 16, No. 1, January/February 2001.

[20] Borges, L.M.S., Falbo, R.A., Managing Software Process Knowledge, *Proc.of CSITeA'2002*, Junho de 2002.

[21] Broomé, M., Runeson, P., Technical Requirements for the Implementation of an Experience Base,. *Proc. of the 11th International Conference on Software Engineering and Knowledge Engineering*, Germany, 1999.

[22] Falbo, R.A., Guizzardi, G., Duarte, K.C., Natali, A.C.C., Developing Software for and with Reuse: An Ontological Approach, *Proceedings of the CSITeA'2002* (to appear).

[23] Falbo, R.A, Guizzardi, G., Duarte, K.C., "An Ontological Approach to Domain Engineering", *Proc. of 14th Int. Conference on Software Engineering and Knowledge Engineering*, SEKE'02, Ischia, Itália, 2002 (to appear).

[24] Arango, G., Prieto-Díaz, R., Domain Analysis Concepts and Research Directions, in *Domain Analysis and Software Systems Modeling*, *IEEE Computer Society Press*, 1991.

[25] Carvalho, A.L., *SIDER: Um Ambiente de Desenvolvimento de Software Orientado ao Domínio Siderúrgico*, Dissertação de Mestrado, UFES, Vitória, Brasil, Julho de 2002.

[26] Wirfs-Brock, R.J., Johnson, R.E., Surveying Current Research in Object-Oriented Design, Communications of the ACM, Vol. 33, No. 9, September 1990.

[27] Bray, T., Paoli, J., Sperberg-McQueen, C.M., *Extensible Markup Language 1.0*, W3C Recommendation, 1998.

[28] Rabarijoana, A., Dieng, R., Corby, O., Exploitation of XML for Corporate Knowledge Management, *Proceedings of EKAW'99*, p. 373-378, 1999.

[29] Benjamins, V. R., Fensel, D., Pérez, A. G., Knowledge Management through Ontologies, *Proc. of the 2nd International Conference on Practical Aspects of Knowledge Management* (PAKM98), Suíça, 1998.