Apoio à Documentação em um Ambiente de Desenvolvimento de Software

Vanessa B. Nunes, Andrea O. Soares, Ricardo A. Falbo
Mestrado em Informática, Universidade Federal do Espírito Santo
Av. Fernando Ferrari, CEP 29060-900, Vitória - ES – Brasil
falbo@inf.ufes.br

Abstract

Documentation is a time-consuming task that is performed through the whole software process. However, many times, it is neglected, due to several problems. In fact, software tools to support documentation tasks are very important. In this paper, we present the approach defined to deal with documentation integration in ODE, an ontology-based software development environment. This approach includes defining an ontology of software artifact and a software tool that supports documentation in ODE, called XMLDoc. XMLDoc was developed based on the software artifact ontology and uses XML to describe document contents.

1. Introdução.

Um aspecto essencial para a qualidade de um sistema de software é a sua documentação. Artefatos de software são peças fundamentais em um processo de desenvolvimento. Eles são produzidos e consumidos por atividades do processo e, assim, estão presentes em todo o ciclo de vida de desenvolvimento de software. Entretanto, muitas vezes, a documentação é negligenciada em decorrência de fatores como falta de uma política organizacional que valorize essa tarefa, falta de ferramentas para apoiar a documentação e prazos exíguos, o que faz com que desenvolvedores optem por deixar a documentação em segundo plano.

Ferramentas provendo facilidades de documentação permitem agilizar o desenvolvimento de software, realizando algumas tarefas automaticamente, e permitem que modificações sejam mais facilmente realizadas, simplificando a tarefa de manutenção. Entretanto, é difícil encontrar ferramentas nas quais a documentação possa ser feita de forma integrada, customizada e consistente. A maioria das ferramentas não oferece um bom suporte à documentação, de forma que o desenvolvedor se vê, muitas vezes, usando um processador de texto para documentar atividades do processo de software.

Este artigo discute como a documentação é integrada ao processo de software no contexto do ambiente ODE (Ontology-based software Development Environment) [1]. ODE é um ambiente de desenvolvimento de software (ADS) baseado em ontologias, isto é, sua infra-estrutura é fundamentada em ontologias de engenharia de software. A padronização de conceitos provida por uma ontologia permite que a comunicação entre as ferramentas que compõem o ambiente seja aprimorada [1]. Dentre as ontologias que compõem a base ontológica de ODE, duas delas merecem destaque no contexto deste trabalho: a ontologia de processo de software [2] e a ontologia de artefato de software. Essas ontologias, que estão integradas, definem os principais conceitos relacionados a artefatos de software e formam a base para a integração da documentação em ODE. De fato, a ontologia de artefato desenvolvida é subdividida em ontologias mais específicas, a saber: ontologia de documento, ontologia de diagrama e ontologia de artefato de código. A ontologia de documento considera o fato de documentos normalmente não se apresentam apenas constituídos de descrições textuais contínuas e sim através de uma estrutura bem definida, composta por seções. Assim, ela trata, também, de aspectos relacionados a padrões organizacionais para a produção de documentos. A ontologia de diagrama tem por base o meta-modelo da UML e descreve os principais elementos que constituem um diagrama. Finalmente, a ontologia de artefatos de código considera as peculiaridades deste tipo de artefato de software, incluindo as linguagens de programação utilizadas para sua elaboração. Devido às limitações de espaço, neste artigo é discutida apenas a ontologia de documentos de software.

Com base na ontologia de documentos, foi desenvolvida XMLDoc, uma ferramenta de apoio à documentação em ODE. XMLDoc visa prover apoio à documentação de forma única e consistente em todo o ambiente, de modo a garantir uma apresentação integrada, usando XML. XMLDoc gera os documentos com base nos dados do repositório central de ODE, favorecendo sua atualização e consistência. Além disso, oferece funcionalidades para apoiar a definição de modelos de documento, o que permite padronização, através da definição da estrutura de tipos de documento. Através da associação dos modelos de documento a folhas de estilo, busca-se garantir integração de apresentação.

Este artigo está estruturado da seguinte forma. A seção 2 discute brevemente o tema documentação e sua relação com ADSs. A seção 3 apresenta brevemente o ambiente ODE e a ontologia de documentos de software utilizada pelo ambiente. Na seção 4, é apresentada XMLDoc, a ferramenta de apoio à documentação de ODE. A seção 5 discute trabalhos correlatos e apresenta as conclusões desse trabalho.

2. Documentação e Ambientes de Desenvolvimento de Software.

A documentação de software é uma atividade essencial no processo de desenvolvimento de software. É através dela que a evolução do software é registrada para que sejam criadas as bases necessárias para as etapas posteriores do processo de software, incluindo treinamento, utilização e manutenção de software [3]. Muitos tipos de documentos são produzidos ao longo do processo de software, tais como propostas, planos de projeto, especificações de requisitos e de projeto, dentre outros. Assim, é comum uma organização gastar de 20 a 30% de todo o esforço de desenvolvimento na elaboração de documentos [4]. Os documentos, por sua vez, têm de apresentar qualidade, para que atividades de avaliação e manutenção possam ser realizadas com sucesso [3]. Entretanto, diversos fatores, como prazos apertados, altos custos, imprecisão e dificuldade na manipulação de documentos, podem comprometer o processo de documentação, levando a documentos incompletos, desatualizados e inconsistentes. De fato, deve-se considerar que a elaboração de uma documentação de qualidade é tão importante quanto a qualidade do software em si [3].

Neste contexto, é importante o uso de ferramentas para apoiar o processo de documentação de software. Tais ferramentas, idealmente, deveriam apoiar todas as atividades do processo de software, já que todas elas envolvem algum tipo de documentação. Dentre as ferramentas de documentação mais utilizadas, destacam-se os editores de textos e os editores de hipertextos. Outras ferramentas incluem extratores de documentação, que extraem documentos de outros artefatos, tal como código, e compositores de documentação, que geram documentos a partir, por exemplo, de modelos construídos em uma ferramenta CASE de modelagem.

Um problema bastante comum no uso de ferramentas de documentação é o pouco suporte oferecido ao processo de documentação em si, incluindo a identificação dos documentos e suas relações com as atividades do processo e padrões organizacionais, o projeto da estrutura dos documentos, incluindo a definição de modelos de documento, e a própria elaboração de documentos. Ferramentas de documentação integradas a um Ambiente de Desenvolvimento de Software (ADS), contudo, podem atenuar alguns desses problemas. Um ADS pode ser visto como um conjunto de ferramentas integradas que facilitam a realização de atividades da engenharia de software, apoiando todo o ciclo de vida do produto de software desenvolvido [5]. Um ADS deve suportar a definição de processos de software e se utilizar desta definição para estabelecer uma ligação explícita entre as ferramentas do ambiente e os processos definidos, incluindo as atividades do processo e os artefatos gerados e consumidos por elas. Assim, é natural pensar a integração da documentação em um ADS. Em um ADS, a identificação da documentação necessária e a relação dos artefatos com outros ativos do processo de software, principalmente atividades, ferramentas e procedimentos, são tratadas na própria definição do processo no ambiente, cabendo à ferramenta de documentação tratar aspectos mais específicos do processo de documentação, tais como a definição de padrões para a elaboração de documentos e o apoio à elaboração em si dos documentos a partir desses padrões. Esta é a abordagem do ambiente ODE.

3. ODE: Um Ambiente de Desenvolvimento de Software Baseado em Ontologias.

ODE (*Ontology-based software Development Environment*) [1] é um ADS Centrado em Processo, que tem sua fundamentação baseada em ontologias. A premissa do projeto de ODE é a seguinte: se as ferramentas de um ADS são construídas baseadas em ontologias, a integração dessas ferramentas pode ser facilitada, pois os

conceitos envolvidos estão bem definidos nas ontologias. Essa é uma das principais características que distingue ODE de outros ADSs: sua base ontológica.

Uma ontologia é um artefato de engenharia, constituído por um vocabulário específico usado para descrever uma certa realidade, mais um conjunto de suposições explícitas referentes ao significado pretendido para os termos usados no vocabulário. Esse conjunto de suposições tem, usualmente, a forma de uma teoria lógica de primeira ordem, onde os termos do vocabulário aparecem como nomes de predicados unários ou binários, respectivamente chamados de conceitos e relações. No caso mais simples, uma ontologia descreve uma hierarquia de conceitos relacionados (taxonomia); em casos mais complexos, axiomas são adicionados para expressar outros relacionamentos entre conceitos e para restringir a interpretação pretendida [6].

A arquitetura de ODE reflete sua base ontológica, sendo composta por dois níveis: o Nível Base e o Meta-Nível. O nível base define as classes que controlam as tarefas realizadas no ambiente e suas ferramentas. O meta-nível, ou pacote *Conhecimento*, define as classes que descrevem o conhecimento sobre os objetos do nível base. As classes do pacote *Conhecimento* são derivadas diretamente de ontologias e seus objetos podem ser vistos como itens de instanciação de uma ontologia. Elas constituem o conhecimento do ambiente, que pode ser utilizado por todas as ferramentas que o compõem [7].

No contexto da integração da documentação em ODE, as ontologias de processo de software [2] e de artefato de software desempenham um importante papel. A primeira define os principais conceitos relacionados a processos de software e é a base para a integração de processo em ODE [7]. A segunda, por sua vez, define os principais aspectos relacionados à documentação de software e é a base para a integração da documentação em ODE. Uma vez que um artefato é um ativo do processo de software, este conceito está presente tanto na ontologia de processo de software [2] quanto na ontologia de artefatos de software. De fato, a ontologia de artefato foi construída integrada à de processo.

O principal objetivo de uma ontologia de artefato de software é favorecer o compartilhamento e o reuso dos diversos artefatos produzidos em um processo de software. Deve-se observar, no entanto, que não é razoável construir uma ontologia única para artefatos de software, uma vez que, para cada tipo de artefato (documento, diagrama, artefato de código, componente de software, dentre outros), pode-se vislumbrar características diferentes. Desta forma, partindo-se da ontologia de processo de software desenvolvida em [2], que define aspectos relacionados a artefatos de forma geral, foram desenvolvidas ontologias específicas para alguns tipos de artefatos, a saber, documento, diagrama e artefato de código. Devido a limitações de espaço, a seguir é apresentada apenas a ontologia de documentos. Essa ontologia foi desenvolvida usando a abordagem sistemática para construção de ontologias descrita em [2], que preconiza o uso de LINGO como linguagem de modelagem de ontologias e lógica de primeira ordem para a sua formalização. A figura 1 mostra a notação principal de LINGO, juntamente com os axiomas impostos pela notação Todo-Parte

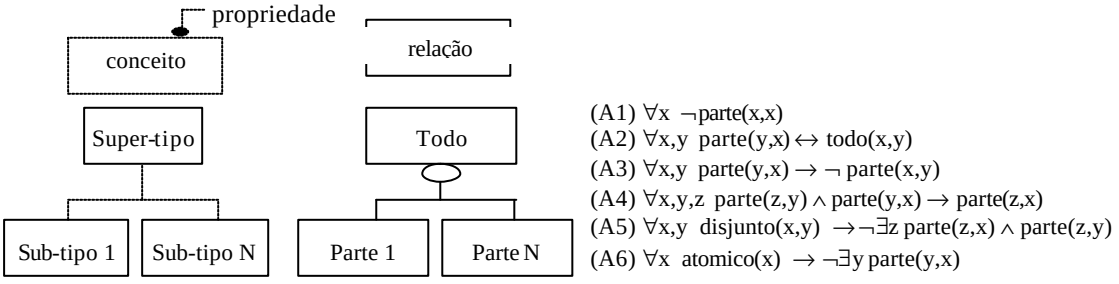

Figura 1 – Parte da Notação de LINGO e Axiomas Impostos pela Relação Todo-Parte.

Para tratar os aspectos envolvidos na documentação de software, a ontologia de documentos deve ser capaz de responder às seguintes questões de competência:

1. A quais documentos um roteiro se aplica?
2. Qual a estrutura definida por um modelo de documento?
3. Qual a estrutura de um documento?
4. Um documento está aderente a um modelo de documento?

A figura 2 mostra o modelo em LINGO da ontologia de documento. Nesse modelo, os conceitos Artefato, Procedimento e Roteiro foram importados da ontologia de processo de software, indicando a integração entre essas ontologias. Vale observar, ainda, que as notações todo-parte preenchidas indicam composição, enquanto as não preenchidas indicam agregação. Os axiomas da figura 1, porém, aplicam-se a todas as relações todo-parte.

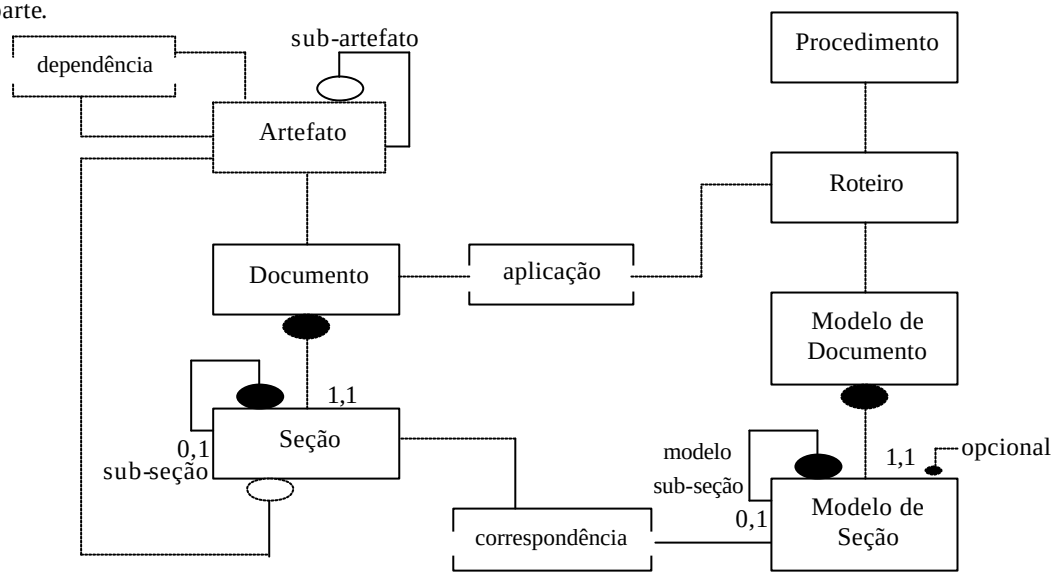

Figura 2 – A Ontologia de Documentos de Software.

Documentos são artefatos de software não passíveis de execução, constituídos tipicamente de declarações textuais, normalmente associados a padrões organizacionais (roteiros) que definem a forma como eles devem ser produzidos. Exemplos de documentos incluem: documento de especificação de requisitos, plano de projeto, plano de qualidade, relatório de avaliação da qualidade, entre outros.

A relação de dependência entre artefatos é transitiva, ou seja, se um artefato *a1* depende de um artefato *a2* e este depende de um terceiro artefato *a3*, então o primeiro (*a1*) depende do terceiro (*a3*):

$$(\forall a1, a2, a3)\ (dependencia(a1, a2) \wedge dependencia(a2, a3) \rightarrow dependencia(a1, a3))$$

É importante destacar, ainda, que se um artefato *a1*, parte do artefato *a2*, depende de outro artefato *a3*, então *a2* também dependem de *a3*. Além disso, um artefato-todo depende sempre de suas partes.

$$(\forall a1, a2, a3)\ (subartefato(a1, a2) \wedge dependencia(a1, a3) \rightarrow dependencia(a2, a3))$$

$$(\forall a1, a2)\ (subartefato(a1, a2) \rightarrow dependencia(a2, a1))$$

Documentos normalmente não se apresentam apenas constituídos de descrições textuais contínuas e sim através de uma estrutura bem definida, composta por seções. Seções, por sua vez, são tipicamente constituídas de declarações textuais, de outras seções e/ou de outros artefatos. É importante destacar que, se uma seção pertence a um determinado documento, suas sub-seções também pertencem a tal documento.

$$(\forall s1, s2, d)\ (partedoc(s1, d) \wedge subseção(s2, s1) \rightarrow partedoc(s2, d))$$

Se um artefato compõe uma seção de um documento, então ele é um sub-artefato desse documento.

$$(\forall s, d, a)\ (partedoc(s, d) \wedge partesec(a, s) \rightarrow subartefato(a, d))$$

Além disso, se um artefato compõe uma seção, ele também compõe as super-seções dessa seção.

$$(\forall a, s1, s2)\ (partesec(a, s1) \wedge subsecão(s1, s2) \rightarrow partesec(a, s2))$$

Roteiros são utilizados como diretrizes para a elaboração de documentos, provendo padronização e facilitando a interpretação de tais documentos. Um modelo de documento é um tipo especial de roteiro que estabelece a estrutura de um documento e instruções para sua elaboração. Modelos de documentos são compostos de modelos de seção, que definem instruções para a elaboração de seções de documentos aderentes

ao respectivo modelo de documento. Um modelo de seção pode ser opcional ou obrigatório. Um modelo de seção opcional é aquele que não necessita de uma seção correspondente nos documentos que aplicam o modelo de documento ao qual pertence, enquanto um modelo de seção obrigatório necessita de uma seção correspondente para ele. Da mesma forma que as seções, um modelo de seção pode ser decomposto em outros modelos de seções.

Apesar dos modelos de documentos serem roteiros que estabelecem a estrutura de um documento e instruções para sua elaboração, nada garante que um documento criado com base em um determinado modelo de documento esteja aderente a ele. Para que um documento esteja aderente a um modelo de documento, deve possuir uma seção correspondente para cada modelo de seção obrigatória do modelo de documento.

(" d, md) aderente(d, md) « ((" ms)(partemodelo(ms, md) Ù Ø opcional(ms)) ®
(($ s) (partedoc(s, d) Ù correspondencia(s, ms))

4. XMLDoc: A Ferramenta de Apoio à Documentação de ODE.

A documentação é uma atividade de apoio que ocorre ao longo de todo o processo de software. De fato, todas as atividades do processo de software são documentadas de alguma forma. Assim sendo, o objetivo principal de XMLDoc é fornecer um meio padronizado de documentação para o ambiente ODE, de modo que não haja necessidade de cada ferramenta isoladamente ter de oferecer alguma funcionalidade de documentação específica para a atividade por ela apoiada.

Alguns dos principais requisitos de XMLDoc incluem: (i) Os desenvolvedores devem ter acesso a documentos atualizados e consistentes; (ii) Deve-se utilizar modelos para estabelecer como os documentos devem ser elaborados, de forma a auxiliar os desenvolvedores na realização dessa atividade e assegurar que questões importantes sejam sempre tratadas; (iii) Documentos e diagramas devem ser consistentes entre si.

Uma tecnologia que tem merecido destaque neste contexto é a tecnologia XML (*eXtensible Markup Language* - Linguagem de Marcação Extensível). XML promete ser um padrão de nível intermediário que pode permitir a criação de ferramentas de software mais flexíveis e reusáveis, resolvendo diversos problemas de integração entre ferramentas [5], dentre eles os problemas de integração de documentação.

Usando a tecnologia XML, XMLDoc gera os documentos sempre com base nos dados do repositório central, favorecendo a atualização e a consistência do documento. A criação de documentos é dinâmica, ou seja, um documento só é efetivamente gerado no instante em que é solicitada a exibição do mesmo. Esta característica é importante, pois garante que o documento está sempre atualizado. Além disso, XMLDoc permite cadastrar modelos de documento. Estes modelos são a base para a padronização da documentação, através da definição da estrutura dos diversos tipos de documentos, e, através da associação destes modelos a uma folha de estilo, favorecem integração de apresentação. Assim, é possível apoiar a elaboração de documentos a partir dos modelos de documento definidos, facilitando a realização desta atividade.

A figura 3 mostra as principais classes de domínio de XMLDoc. Vale ressaltar que a arquitetura em dois níveis de ODE é preservada, ou seja, as classes do pacote *Documentação* dizem respeito aos documentos produzidos em um projeto de software específico. As classes do pacote *Conhecimento* mostradas referem-se ao conhecimento sobre os artefatos e procedimentos, estes últimos especializados para tratar modelos de documentos e sua estrutura. Em ambos os casos, é fácil perceber que esses modelos foram construídos com base na ontologia de artefato de software, tendo sido adicionados alguns atributos e associações, específicos para tratar requisitos de XMLDoc, tais como o atributo *folhaEstilo* em *ModeloDocumento* e os atributos *ordem* e *instruções* em *ModeloSeção*.

5. Conclusão.

Existem muitas ferramentas de apoio à documentação disponíveis, inclusive comercialmente. Pressman [4] no *site* de seu livro (www.mhhe.com/engcs/compsci/pressman/ole_linkedcontent/casetools.html) aponta para diversas delas, tais como JVision, Cradle, SoDA e DocExpress. Analisando essas e outras ferramentas, pode-se notar que, do ponto de vista de elaboração de documentos, XMLDoc não apresenta nenhuma funcionalidade nova, apenas procura disponibilizar aquelas que foram consideradas as mais importantes.

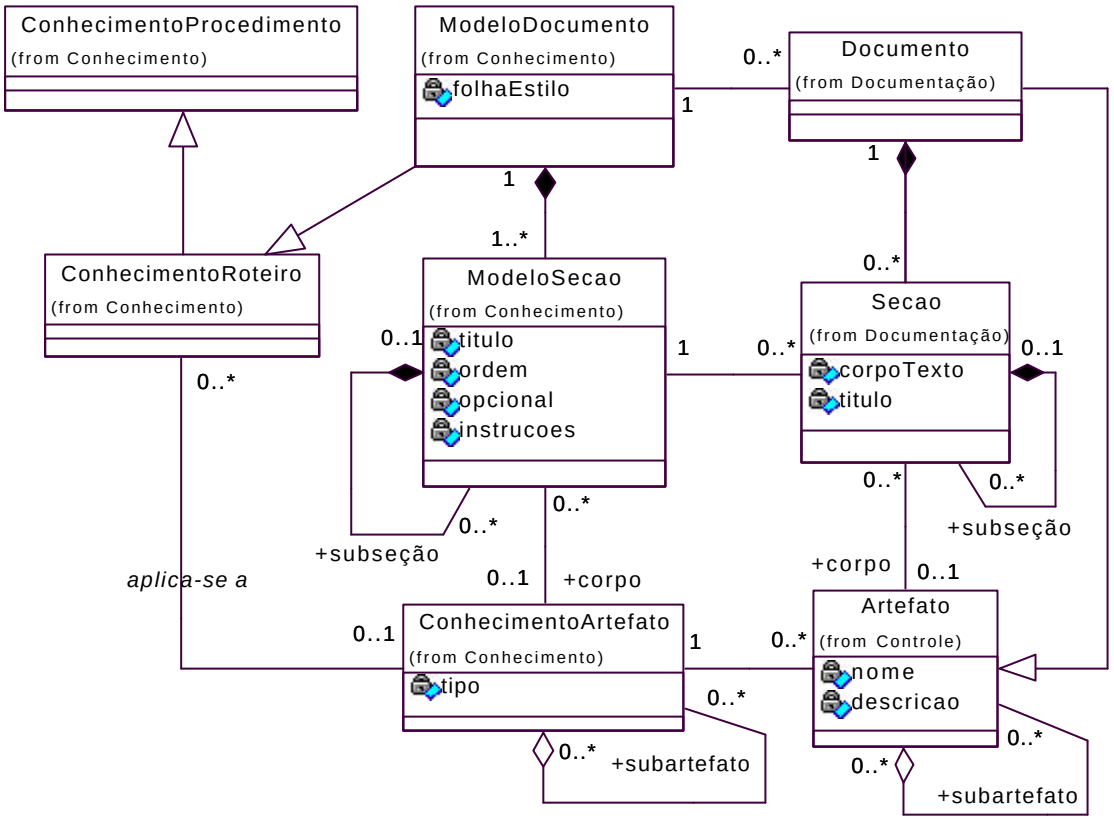

Figura 3 – Modelo de Classes de XMLDoc.

As características marcantes de XMLDoc são: (i) ser baseada em uma ontologiaç (ii) estar integrada a um ADS e, por conseguinte, se utilizar de toda a funcionalidade de controle de processos de software disponível no ambiente. Deste modo, a identificação dos documentos e suas relações com as atividades do processo e padrões organizacionais é estabelecida na definição de processos de ODE [7].

5. Agradecimentos.

Os autores agradecem ao CNPq pelo apoio financeiro a este trabalho.

6. Referências.

[1] 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, La Plata, Argentina, 2003, pp 931-940.

[2] R.A. Falbo, A.R.C. da Rocha, C.S. Menezes, "A Systematic Approach for Building Ontologies". In: Proceedings of the 6th Ibero-American Conference on Artificial Intelligence, Lisbon, Portugal, 1998.

[3] R. Sanches, "Documentação de Software", In: *Qualidade de Software: Teoria e Prática*, Prentice Hall, São Paulo, 2001, pp 54-59.

[4] R.S. Pressman. *Engenharia de Software*, 5a edição, McGraw-Hill, Rio de Janeiro, 2002.

[5] Harrison, W.; Ossher, H.; Tarr, P. "Software Engineering Tools and Environments: A Roadmap", In: Proc. of The Future of Software Engineering, ICSE'2000, Limerick, Ireland, 2000.

[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] F.B. Ruy, G. Bertollo, R.A. Falbo, "Apoio Baseado em Conhecimento à Integração de Processos em ODE", In: Proc. 3rd JIISIC'2003, Valdivia, Chile, 2003.