Uma Ferramenta de Gerência de Configuração Integrada a um Ambiente de Desenvolvimento de Software

Article · January 2006

CITATIONS
3

READS
42

2 authors:

Vanessa Battestin Nunes
Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo...
35 PUBLICATIONS **47** CITATIONS
SEE PROFILE

Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo
172 PUBLICATIONS **1,661** CITATIONS
SEE PROFILE

Uma Ferramenta de Gerência de Configuração Integrada a um Ambiente de Desenvolvimento de Software

Vanessa Battestin Nunes, Ricardo de Almeida Falbo

Mestrado em Informática – Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari s/n, Campus de Goiabeiras – 29.060-900 – Vitória – ES – Brasil
vanessabattestin@yahoo.com.br, falbo@inf.ufes.br

Abstract. *This paper presents the software configuration management (SCM) tool of the software engineering environment ODE (Ontology-based software Development Environment). Since tool integration in a software engineering environment is a complex problem, it discusses important features that the SCM tool must present in order to facilitate integration. Furthermore, as long as ODE is based on ontologies, an ontology of SCM was developed, and used as basis for developing the tool.*

Resumo. *Este artigo apresenta a ferramenta de gerência de configuração de ODE, um Ambiente de Desenvolvimento de Software (ADS) baseado em ontologias. Dado que a integração de ferramentas em ADSs é um problema complexo, o artigo discute características importantes da ferramenta que visam a facilitar a integração. Além disso, uma vez que ODE é baseado em ontologias, uma ontologia de gerência de configuração foi desenvolvida e usada como base para a construção da ferramenta.*

1. Introdução

Ambientes de Desenvolvimento de Software (ADSs) podem ser definidos como coleções de ferramentas integradas que facilitam as atividades da engenharia de software, durante todo o ciclo de vida do software ou, pelo menos, em porções significativas dele [Harrison et al., 2000]. Uma vez que as ferramentas têm de trabalhar harmoniosamente para apoiar os profissionais de software, a questão da integração de ferramentas em ADSs é essencial para o sucesso da automatização do processo.

Integração em ADSs envolve diversas dimensões, tais como dados, controle, conhecimento, apresentação e processo [Pfleeger, 2004], [Falbo et al., 2003]. Em outras palavras, as ferramentas têm de ser capazes de compartilhar dados, funcionalidades e conhecimento, têm de apresentar interfaces homogêneas, e sua ativação e interação têm de ser realizadas respeitando-se um processo de software previamente definido para o projeto no ADS, no caso dito um ADS centrado em processo.

Além disso, é fundamental que os artefatos produzidos no ADS tenham sua configuração gerenciada. Esse é um requisito básico para a qualidade do processo de software e está intimamente ligado à questão da integração. A importância dos sistemas de gerência de configuração de software (GCS) em ADSs é tão reconhecida, que Fuggetta

(2000) aponta que diversos pesquisadores consideram que eles são os verdadeiros ADSs centrados em processo. Pressman (2002) também compartilha essa visão, apontando uma solução para o problema de integração em que o principal mecanismo de integração de ferramentas é a GCS. Ela deve identificar os artefatos a serem gerenciados, prover controle de versão e gerência de mudanças, apoiar auditorias e fornecer relatos de configuração, ficando, assim, no núcleo do ADS.

Este artigo apresenta a ferramenta de gerência de configuração de ODE (*Ontology-based software Development Environment*) [Falbo et al., 2003], discutindo características importantes da mesma, que visam a facilitar a integração. Uma vez que ODE é um ADS baseado em ontologia, uma ontologia de gerência de configuração foi desenvolvida e usada como base para a construção da ferramenta. A seção 2 discute com um pouco mais de detalhes a sinergia entre ADSs e GCS. A seção 3 apresenta a ontologia de GCS desenvolvida. Na seção 4, é apresentada a ferramenta de GCS, discutindo-se os principais aspectos que influenciaram o seu projeto. As seções 5 e 6 apresentam, respectivamente, trabalhos correlatos e as conclusões deste trabalho.

2. Ambientes de Desenvolvimento de Software e Gerência de Configuração

Com o aumento da complexidade dos sistemas de software, passou a ser fundamental o apoio automatizado ao processo de software. Apesar dos benefícios do uso de ferramentas CASE individuais, o número e a variedade de ferramentas usadas passou a ser muito grande, criando dificuldades em seu uso combinado e, muitas vezes, exigindo repetição desnecessária de tarefas. Visando a tratar esses problemas, surgiram os Ambientes de Desenvolvimento de Software (ADSs), buscando integrar ferramentas para apoiar o engenheiro de software na construção de produtos de software, abrangendo várias atividades do ciclo de vida do software.

Entretanto, a integração de ferramentas em ADSs não é trivial. Ela envolve, de fato, diversas facetas, tais como:

- Integração de dados: trata do compartilhamento de informações entre ferramentas e é, geralmente, apoiada por serviços de repositório de dados, provendo uma forma única de armazenamento e gerência de objetos;
- Integração de controle: diz respeito ao controle de eventos e ao compartilhamento de funcionalidades, permitindo que ferramentas ativem umas as outras quando necessário;
- Integração de processo: concerne à definição de processos e ao uso de processos definidos no ambiente para controlar o acesso a ferramentas e artefatos, estabelecendo uma ligação clara entre as ferramentas e o processo de software;
- Integração de apresentação: refere-se à apresentação uniforme das interfaces com o usuário do ambiente e de suas diversas ferramentas, aumentando a usabilidade;
- Integração de conhecimento: trata do compartilhamento de conhecimento (incluindo experiências) e do suporte baseado em conhecimento nas ferramentas, demandando serviços de gerência de conhecimento e de inferência.

Dentre essas dimensões de integração, a integração de dados assume um caráter especialmente importante, sendo, em muitas situações, a base para a integração em ADSs. Para se conseguir integração de dados, é fundamental integrar serviços de gerência de configuração de software ao ambiente. Durante o processo de software, vários artefatos são produzidos e alterados, evoluindo até que seus propósitos sejam alcançados. Caso essas alterações não sejam devidamente documentadas e comunicadas, diversos problemas poderão acontecer, tais como dois ou mais desenvolvedores estarem alterando um mesmo artefato ao mesmo tempo, não se saber qual a versão mais atual de um artefato etc. Para evitar esses problemas, é de suma importância o acompanhamento e controle de artefatos por meio de um processo de gerência de configuração, durante todo o ciclo de vida do software [Sanches, 2001].

A Gerência de Configuração de Software (GCS) visa a controlar a evolução dos sistemas de software e deve identificar e documentar os artefatos que podem ser modificados, estabelecer as relações entre eles e os mecanismos para administrar suas diferentes versões ou variantes, controlar as modificações, além de fazer auditoria e preparar relatórios sobre tais modificações [Pressman, 2002], [SWEBOK, 2001].

O processo de GCS se inicia com a confecção de um plano de gerência de configuração, identificando os itens que serão colocados sob gerência de configuração, chamados itens de configuração. Deve-se descrever, ainda, como eles se relacionam, isto é, qual a dependência entre eles. Isso é muito importante para as futuras manutenções, pois permite identificar de maneira eficaz os itens afetados em decorrência de uma alteração [SWEBOK, 2001], [Sanches, 2001]. Além disso, deve ser criado um esquema de identificação dos itens de configuração, com atribuição de nomes exclusivos, para que seja possível estabelecer a evolução de cada versão ou variante dos itens. Neste texto, versões e variantes de itens de configuração são tratados genericamente como variações de itens de configuração.

Uma vez identificados os itens de configuração, devem ser planejadas as linhas-base dentro do ciclo de vida do projeto. Uma linha-base (ou *baseline*) é uma versão estável de um sistema contendo todos os componentes que constituem este sistema em um determinado momento. Nos pontos estabelecidos pelas linhas-base, os itens de configuração devem ser identificados, analisados, corrigidos, aprovados e armazenados em um local sob controle de acesso, denominado repositório central. A partir desse momento, qualquer alteração em um item só poderá ser realizada por meio de procedimentos formais de controle de alteração [Sanches, 2001], [Pressman, 2002]. Em contrapartida, os itens não colocados sob GCS podem ser alterados livremente.

O passo seguinte do processo de GCS é o controle de versão, que visa identificar, armazenar e administrar diferentes variações dos itens de configuração [Sanches, 2001], [Pressman, 2002]. A idéia é que, a cada alteração realizada em um item de configuração, uma nova versão ou variante seja criada [Estublier, 2000]. Versões de um item são as revisões geradas pelas diversas alterações, enquanto variantes são as diferentes formas de um item, que existem simultaneamente e atendem a requisitos similares [Sanches, 2001].

Outra, e talvez a mais importante, atividade do processo de GCS é o controle de alteração, que combina procedimentos humanos e ferramentas automatizadas para controlar as alterações realizadas nos itens de configuração [Pressman, 2002]. Assim que uma alteração é solicitada, o impacto em outros itens de configuração e o custo para a modificação devem ser avaliados [SWEBOK, 2001]. Um responsável deve decidir se a alteração poderá ou não ser realizada. Caso a alteração seja liberada, pessoas são indicadas para a sua execução. Assim que não houver ninguém utilizando os itens de configuração envolvidos, eles poderão ser retirados do repositório central para alteração, em um procedimento denominado *check-out*. A partir desse momento, nenhum outro desenvolvedor poderá alterar esses itens. Os desenvolvedores designados fazem as alterações necessárias e, assim que essas forem concluídas, os itens são submetidos a uma revisão. Se as alterações forem aprovadas, os itens são devolvidos ao repositório central, estabelecendo uma nova linha-base, em um procedimento chamado *check-in* [Sanches, 2001], [Pressman, 2002].

Deve-se observar que, mesmo com mecanismos de controle bem estabelecidos, não é possível garantir que as modificações foram corretamente implementadas. Assim, revisões técnicas formais e auditorias de configuração de software são necessárias no processo de GCS [Pressman, 2002]. Essas atividades tentam descobrir omissões ou erros na configuração e verificar se procedimentos, padrões, regulamentações ou guias foram devidamente aplicados no processo e no produto [Sanches, 2001], [SWEBOK, 2001].

Por fim, o último passo do processo de GCS é a preparação de relatórios, visando relatar a todas as partes envolvidas o estado de configuração dos itens.

Existem diversos sistemas de GCS, tais como CVS (*Concurrent Version System*) [Caetano, 2004] e Rational ClearCase [Wahli et al., 2004], muitos deles pautados sobre um esquema de arquivos. Entretanto, em um ADS, esse pode não ser o esquema mais interessante, sobretudo quando o repositório central do ADS é implementado como um banco de dados. É clara a relação entre a GCS e a dimensão de integração de dados em um ADS. Mas, ao se integrar a GCS em um ADS, as demais dimensões de integração anteriormente citadas também devem ser exploradas. No que se refere, à dimensão de apresentação, a ferramenta de GCS deve apresentar interfaces homogêneas em relação ao restante do ambiente. No que concerne à integração de controle, é importante que as várias ferramentas, usadas para produzir artefatos, sejam capazes de ativar ou de se apoiar em serviços da GCS para que seja possível fazer um controle efetivo dos artefatos. Já no que tange à integração de conhecimento, há de se considerar que artefatos são tratados pela gerência de conhecimento, na grande maioria das vezes, como importantes itens formais de conhecimento [Natali & Falbo, 2003]. Por fim, no que tange à integração de processo, ao se definir um processo de software no ADS, deve-se apontar a GCS como um processo contínuo ao longo do ciclo de vida.

Considerando que o uso de um sistema isolado de GCS é insuficiente para atender a todas essas considerações, no contexto do Projeto ODE (*Ontology based software Development Environment*) [Falbo et al., 2003], foi desenvolvida uma ferramenta integrada de GCS, procurando considerar os diversos aspectos inerentes à integração em ADSs. Uma vez que ODE é um ADS baseado em ontologias, foi desenvolvida uma ontologia de GCS, apresentada a seguir.

3. Uma Ontologia de Gerência de Configuração de Software

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, relações e suas propriedades em algum domínio, e um conjunto de axiomas, que restringem a sua interpretação [Guarino, 1998]. 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 [Davies et al., 2003].

O uso de ontologias pode trazer muitos benefícios para a integração de ferramentas em ADSs. Se as ferramentas de um ADS são construídas baseadas nas mesmas ontologias, elas compartilham uma conceituação, reduzindo confusões terminológicas e facilitando o entendimento e a comunicação entre as ferramentas e as pessoas que as utilizam [Falbo et al., 2003].

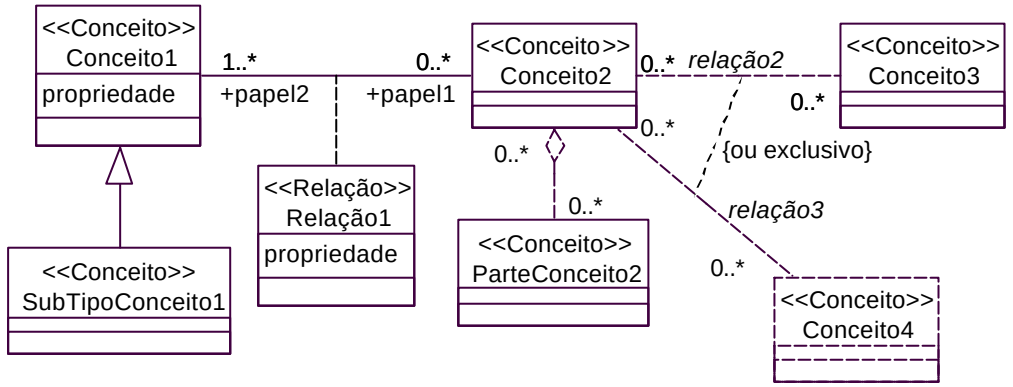
Tendo em vista que os artefatos são a base para a integração de dados e, portanto, o meio mais direto de comunicação entre as ferramentas, é importante definir uma ontologia de artefatos de software, de modo que todas as ferramentas tenham um entendimento compartilhado sobre esse universo de discurso. Dada a diversidade de tipos de artefatos, sub-ontologias podem ser definidas para esses tipos (p.ex., diagrama, documento, código-fonte etc), adicionando outros elementos e garantindo que cada sub-ontologia tem um grau de comprometimento ontológico mínimo. Além disso, uma vez que, artefatos devem ser submetidos à GCS, uma ontologia tratando aspectos relacionados a esse importante processo também deve ser considerada.

Assim, no contexto do projeto ODE, optou-se por desenvolver um conjunto de ontologias para tratar de maneira ampla o domínio de artefatos de software e da GCS. O ponto de partida para essa ontologia foi a ontologia de processo de software desenvolvida em [Falbo, 1998], que define aspectos relacionados a artefatos de forma geral, no que se refere às suas interações com as atividades do processo de software. Integrada à ontologia de processo, foi desenvolvida uma ontologia básica de artefatos de software e, a partir dessa, foram desenvolvidas sub-ontologias específicas para alguns tipos de artefatos, a saber, documento [Nunes et al., 2004], diagrama e artefato de código. Para maiores detalhes sobre o conjunto das ontologias de artefato de software, vide [Nunes, 2005]. Paralelamente, definiu-se uma ontologia de GCS, dada a grande importância desse processo. Essa ontologia é apresentada neste artigo.

Para desenvolver a ontologia de GCS, foi adotado o método SABiO (***S****ystematic* ***A****pproach for* ***Bui****lding* ***O****ntologies*) [Falbo, 2004]. SABiO define um processo para construção de ontologias, cujas principais atividades são: (i) identificação do propósito e especificação de requisitos, que visa a identificar questões que a ontologia deve ser capaz de responder (questões de competência), (ii) captura da ontologia, que tem por objetivo capturar os conceitos, relações, propriedades e restrições relevantes sobre o domínio em questão; (iii) formalização, que busca escrever os axiomas da ontologia em uma linguagem formal e (iv)

avaliação da ontologia, que, dentre outros, trata de avaliar se a ontologia é capaz de responder às questões de competência.

Em sua versão mais recente, SABiO advoga o uso de um perfil da UML como linguagem gráfica para representação de ontologias. Esse perfil utiliza um subconjunto de elementos da UML exercendo o mesmo papel da notação de LINGO [Falbo, 1998], a linguagem originalmente proposta. LINGO possuía primitivas para representar conceitos, relações e propriedades, e alguns tipos de relações que possuíam uma semântica bem-estabelecida, tais como relações todo-parte e sub-tipo-de, para os quais um conjunto de axiomas formais, ditos axiomas epistemológicos, era definido. Assim, apesar de se utilizar os elementos de modelo da UML, a semântica imposta é a mesma que a estabelecida para os correspondentes elementos em LINGO, conforme parcialmente mostrado na figura 1. Segundo esse perfil da UML, classes com estereótipo `<<Conceito>>` representam conceitos da ontologia. Relações são definidas como associações nomeadas. Propriedades de conceitos e relações são representadas como atributos de classes. Relações que contêm propriedades ou que possuem aridade maior que dois são representados como classes associativas com estereótipo `<<Relação>>`. Relações de supertipo e todo-parte são representadas como relações de generalização/ especialização e de agregação/composição, respectivamente. Finalmente, condicionantes entre relações são representados por restrições entre associações [Mian, 2003].



Axiomas:
Todo-Parte
(AE1) ∀x ¬parte(x,x)
(AE2) ∀x,y parte(y,x) ↔ todo(x,y)
(AE3) ∀x,y parte(y,x) → ¬ parte(x,y)
(AE4) ∀x,y,z parte(z,y) ∧ parte(y,x) → parte(z,x)
(AE5) ∀x,y disjunto(x,y) → ¬∃z parte(z,x) ∧ parte(z,y)
(AE6) ∀x atomico(x) → ¬∃y parte(y,x)

Sub-tipo:
(AE7) (∀x,y,z) (subtipo(x,y)∧subtipo(y,z)→ subtipo(x,z))
(AE8) (∀x,y) (subtipo(x,y)→supertipo(y,x))
Condicionante Ou-exclusivo:
(AE9) (∀a∈C2) ((∃b) (b∈C3)∧R2(a,b))→ ¬((∃c∈C4)∧R3(a,c)))
(AE10) (∀a∈C2)((∃c) (c∈C4)∧R3(a,c)) →

Figura 1 – Notações da extensão da UML para representar ontologias e seus axiomas associados [Mian, 2003].

Tomando por base o escopo da GCS, foram levantadas as seguintes questões de competência para a ontologia de GCS:

1. Que itens estão sob gerência de configuração?
2. Quais as variações (versões/variantes) de um item de configuração?
3. Como a variação de um item de configuração se decompõe?

4. Em quais outras variações uma alteração de uma determinada variação poderá provocar impactos?
5. Uma variação de um item de configuração derivado de um artefato está aderente à estrutura (decomposição e dependências) de tal artefato?
6. Quais as variações sujeitas a modificação em uma determinada alteração?
7. Quais as variações produzidas por uma determinada alteração?
8. Que variações de itens de configuração compõem uma determinada linha base?
9. A quais itens de configuração um determinado recurso humano tem acesso? E qual é o tipo de acesso a ele liberado?
10. Quem é o responsável por uma determinada alteração?
11. Uma determinada variação de um item de configuração está disponível para ser alterada?

Analisando as questões de competência anteriormente relacionadas, os seguintes aspectos foram considerados na captura da ontologia de GCS: itens sob gerência de configuração, variações de itens de configuração, decomposição e dependência entre variações, alteração de variações, formação de linha-base, responsabilidade de alteração e acesso a itens de configuração. Para tratar esses aspectos, foram elaborados um modelo gráfico, um dicionário de termos e um conjunto de axiomas escritos em lógica de primeira ordem. Conforme apontado anteriormente, a ontologia de GCS foi desenvolvida integrada às ontologias de processo de software [Falbo, 1998] e de artefatos de software [Nunes, 2005].

A figura 2 e a tabela 1 mostram, respectivamente, o modelo gráfico e o dicionário de termos da ontologia de GCS. Além deles, diversos axiomas foram definidos para a ontologia de GCS. O uso do perfil da UML proposto em [Mian, 2003] permite a não apresentação dos axiomas de caráter epistemológico (aqueles derivados simplesmente da estrutura dos conceitos e não de seus significados particulares), já que eles são capturados pela notação. Apenas para ilustrar, a figura 3 apresenta alguns axiomas epistemológicos: (AE1) a (AE6) referem-se à relação todo-parte entre variações; (AE7) resume as restrições impostas pela condicionante ou-exclusivo nas relações de derivação entre item de configuração e artefato e ferramenta de software; (AE8) e (AE9), por sua vez, tratam de conseqüências lógicas da relação sub-tipo.

Além dos axiomas epistemológicos, há outros axiomas, ditos ontológicos, que expressam restrições não capturadas na estruturação dos conceitos e relações. Sejam, por exemplo, as relações de dependência e decomposição de variações. Uma variação pode ser diretamente dependente de outras variações. Contudo, ela pode depender indiretamente em casos como: (i) Se uma variação depende de outra variação, suas super-variações também dependem de tal variação.

(" v1, v2, v3) (dependênciaVariação(v1, v2) Ù superVariação(v3, v1) ® dependênciaVariação(v3, v2))

(ii) Se uma variação *v1* depende de outra variação *v2* que, por sua vez, depende de uma variação *v3*, então *v1* também depende de *v3*.

(" v1, v2, v3) (dependênciaVariação(v1, v2) Ù dependênciaVariação(v2, v3) ® dependênciaVariação(v1, v3))

Ainda neste contexto, quando uma variação composta é submetida a uma alteração, suas sub-variações também o são, como aponta o seguinte axioma:

(" v1, v2, a) (submissão(v1, a) Ù sub-variação(v2, v1) ® submissão(v2, a))

Para um exame completo dos axiomas da ontologia de GCS, vide [Nunes, 2005].

4. Uma Ferramenta de Gerência de Configuração

Tomando por base o universo de discurso tratado na ontologia de GCS, foi desenvolvida uma ferramenta de GCS para o ambiente ODE (*Ontology-based software Development Environment*) [Falbo et al., 2003]. O objetivo principal de ODE é ser um ambiente integrado, fornecendo serviços de infra-estrutura e capaz de integrar ferramentas ao longo das dimensões de integração discutidas na seção 2, usando ontologias para isso.

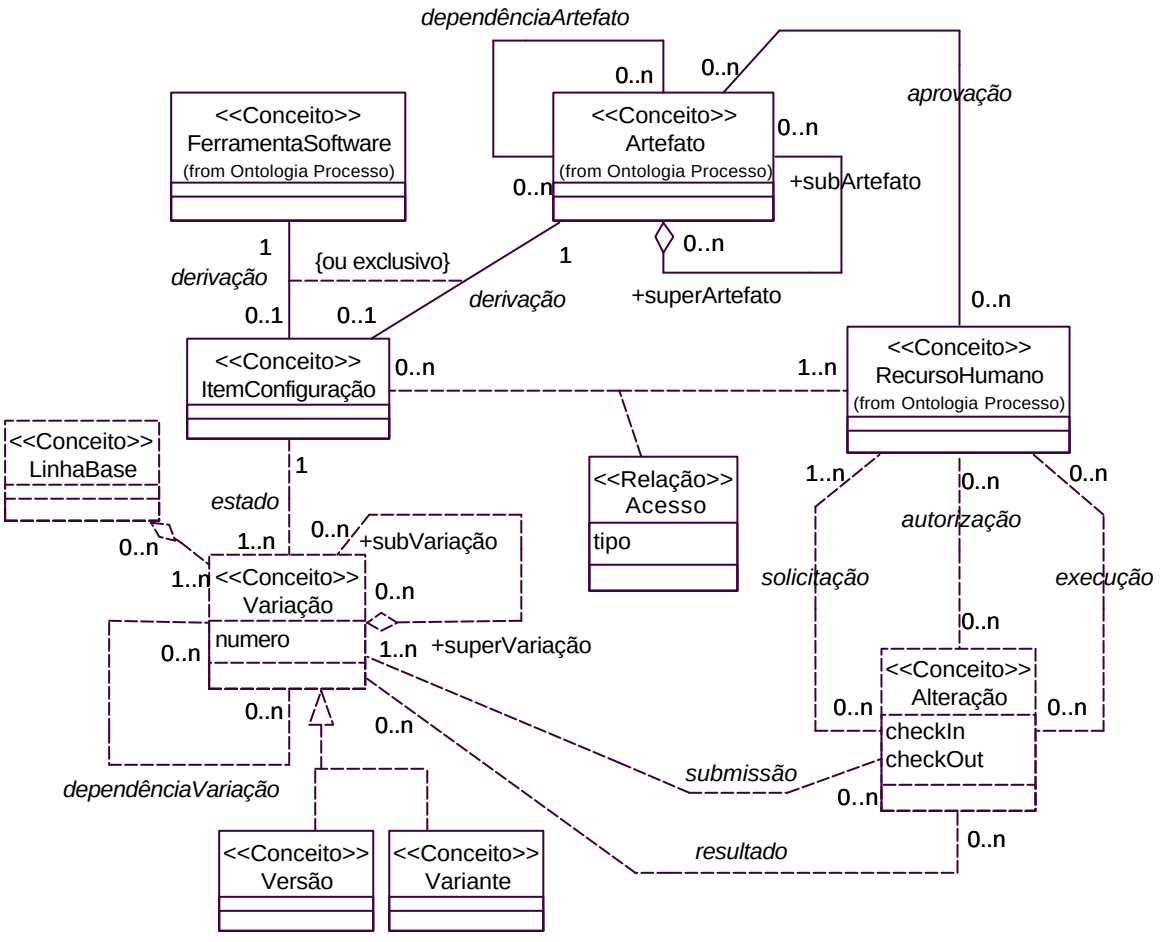


Figura 2 – O Modelo da Ontologia de GCS.

AE1: *(" v) ¬ subVariação(v,v)*
AE2: *(" v1, v2) (subVariação(v1, v2) « superVariação(v2, v1))*
AE3: *(" v1, v2) (subVariação(v1, v2) ® ¬ subVariação(v2, v1))*
AE4: *(" v1, v2, v3) (subVariação(v1,v2) Ù subVariação(v2, v3) ® subVariação(v1, v3))*
AE5: *(" v) (variaçãoElementar(v) « ¬ ($ v1) (subVariação(v1,v)))*
AE6: *(" v) (macroVariação(v) « ¬ ($ v1) (superVariação(v1, v)))*

AE7: *(" x, i) (derivação(i,x) ® itemConfiguração(i) Ù (artefato(x) Ú ferramentaSoftware(x))*
AE8: *(" v) (versão(v) ® variação(v))*
AE9: *(" v) (variante(v) ® variação(v))*

Figura 3 – Alguns Axiomas Epistemológicos da Ontologia de GCS.

ODE está sendo desenvolvido no Laboratório de Engenharia de Software da Universidade Federal do Espírito Santo (LabES/UFES) e desde outubro de 2004 está implantado em uma *software house*, através de uma parceria universidade-empresa. ODE é implementado em uma plataforma livre, que inclui Java e PostgreSQL, rodando sobre o sistema operacional Linux. Em sua versão corrente, ODE possui diversas ferramentas, dentre elas ferramentas de apoio à definição de processos de software, à gerência de riscos, à realização de estimativas, à documentação, à alocação de recursos humanos, à modelagem segundo a UML etc.

Tabela 1 – Dicionário de Termos da Ontologia de GCS.

Acesso	Relação entre **item de configuração** e **recurso humano**, indicando qual **tipo** de acesso (leitura, alteração, exclusão, etc.) determinado **recurso humano** tem sobre determinado **item de configuração**.
Alteração	Representa uma solicitação de alteração efetuada por um **recurso humano**, indicando as **variações de itens de configuração** que poderão ser alteradas. Esta alteração, sendo aprovada, deverá ser executada por **recursos humanos**, dando origem a novas **variações**. Para um controle formal, deverão ser executados procedimentos de ***check-out*** e ***check-in***.
Autorização	Relação entre **recurso humano** e **alteração**, indicando quais **recursos humanos** são os responsáveis por autorizar uma determinada **alteração**.
Check-in	Propriedade de **alteração** que indica quando as **variações** submetidas para uma **alteração** foram devolvidas ao repositório central, estando disponíveis para novas **alterações**. Contém a data e a hora em que as **variações** tornaram-se disponíveis novamente.
Check-out	Propriedade de **alteração** que indica quando cópias das **variações** submetidas para **alteração** foram retiradas do repositório central e disponibilizadas na área de trabalho do desenvolvedor, para serem manipuladas. Contém a data e a hora em que as **variações** foram retiradas do repositório central. A partir deste momento, nenhum outro **recurso humano** poderá alterar estas **variações** até que se tenha realizado um procedimento de ***check-in***.
Dependência entre variações	Relação entre **variações** indicando as **variações** que são dependentes de uma determinada **variação**. Assim, é possível identificar todas as **variações** que podem ser afetadas em uma determinada **alteração** e não apenas as solicitadas.
Derivação	Relação entre **item de configuração** e **artefato** (ou entre **item de configuração** e **ferramenta de software**), indicando que um determinado **artefato** (ou **ferramenta de software**) está sob gerência de configuração, ou seja, tornou-se um **item de configuração**.
Estado	Relação entre **item de configuração** e **variação** que indica quais as **variações** de um certo **item de configuração**.
Execução	Relação entre **recurso humano** e **alteração,** indicando quais **recursos humanos** são responsáveis pela execução de uma determinada **alteração**, ou seja, que **recursos humanos** estão (ou estiveram) de posse de determinadas **variações** para, possivelmente, realizar **alterações**.
Ferramenta de Software	Recurso de software utilizado para apoiar a realização de uma **atividade**. Ex.: Editor de textos, Planilha eletrônica, ferramentas CASE etc.
Item de Configuração	Item que está sob gerência de configuração e, assim, só pode ser alterado segundo um procedimento de controle de alteração formalmente estabelecido e documentado. Pode possuir várias **variações**. Pode representar uma **ferramenta de software** ou um **artefato**, como, por exemplo, um determinado plano de projeto ou um certo artefato de código.
Linha Base	Conjunto de **itens de configuração** em determinadas **variações**, que serve de base para o desenvolvimento ulterior. Ex.: uma linha base formada pela porção de código X (vesão 2.0), pelo documento de plano de projeto Z (variante 1.1.1), pelo diagrama de casos de uso (versão 1.0) etc.
Resultado	Relação entre **variação** e **alteração**, indicando quais **variações** foram produzidas em uma determinada **alteração**, ou seja, quais as **variações** resultantes de uma determinada **alteração**.
Solicitação	Relação entre **recurso humano** e **alteração**, indicando quais **recursos humanos** fizeram a solicitação de uma determinada **alteração**.
Submissão	Relação entre **variação** e **alteração**, indicando quais **variações** foram submetidas para modificação em uma determinada **alteração**.

Tabela 1 – Dicionário de Termos da Ontologia de GCS (continuação).

Sub-variação	Papel da relação de todo-parte entre duas **variações** v_1 e v_2. Se v_2 é parte de v_1, então v_2 é dita uma sub-variação de v_1
Super-variação	Papel da relação todo-parte entre duas **variações** v_1 e v_2. Se v_1 é decomposta em outras **variações**, dentre elas v_2, então v_1 é dita um super-variação de v_2.
Variação	Indica qual a **versão** ou **variante** de um determinado **item de configuração**. É caracterizada por um **número** único para cada **variação** de um mesmo **item de configuração**.
Variante	**Variação** de um **item de configuração** que existe simultaneamente em duas ou mais formas diferentes que atendam a requisitos similares. Ex.: Um documento estava na versão 1.0 e após alterações passou para a versão 1.1. Porém, foi criada a variante 2.0 que, apesar de atender aos mesmos requisitos que a versão 1.1, foi construída de forma diferente.
Versão	Tipo de **variação** de **item de configuração** sem as restrições impostas para **variantes**. Ex.: Versão 1.0 de um documento. Versão 6.0 de uma ferramenta CASE.

Dentre as ferramentas de ODE, duas estão intimamente ligadas à GCS – a ferramenta de apoio à documentação (XMLDoc) [Nunes et al., 2004] e o sistema de gerência de conhecimento do ambiente [Natali & Falbo, 2003] – e, portanto, visando à integração, tiveram grande influência nas decisões a cerca do projeto da ferramenta de GCS. De fato, os seguintes aspectos inerentes ao ambiente ODE foram considerados:

- O ambiente ODE possui um repositório central, implementado como um banco de dados relacional;
- A Gerência de Conhecimento de ODE possui uma memória organizacional, em que os artefatos são itens de conhecimento formais armazenados. Originalmente, todos os artefatos do repositório central de ODE eram considerados itens de conhecimento pela gerência de conhecimento;
- ODE possui uma ferramenta de apoio à documentação, XMLDoc, que utiliza funcionalidade de transformação da representação interna dos artefatos de ODE para documentos XML e exibe esses documentos para os usuários.

A partir da consideração desses aspectos e tomando por base a ontologia de GCS, várias abordagens para a ferramenta de GCS foram consideradas, para se definir qual seria a adotada em ODE, chegando-se, enfim, à abordagem esquematizada na Figura 4, que trata, conjuntamente, o repositório central do ambiente e arquivos XML. Nessa abordagem, os artefatos de software que não estiverem sob GCS e as variações mais atuais dos artefatos sob GCS são armazenados em uma única base de dados relacional, o repositório central de ODE. Além disso, todas as variações dos artefatos sob GCS, incluindo também as mais atuais, são armazenadas na forma de arquivos XML.

Ao ser colocado sob gerência de configuração, um artefato deixa de pertencer ao grupo de artefatos que não estavam sob gerência de configuração e passa a pertencer ao grupo de artefatos sob GCS. Isso é transparente para o ambiente como um todo, uma vez que a base de dados (o repositório central de ODE) continua sendo a mesma. De fato, apenas o artefato passa a ser considerado um item de configuração e, portanto, uma variação (neste caso a versão inicial) é criada, juntamente com seu correspondente arquivo XML.

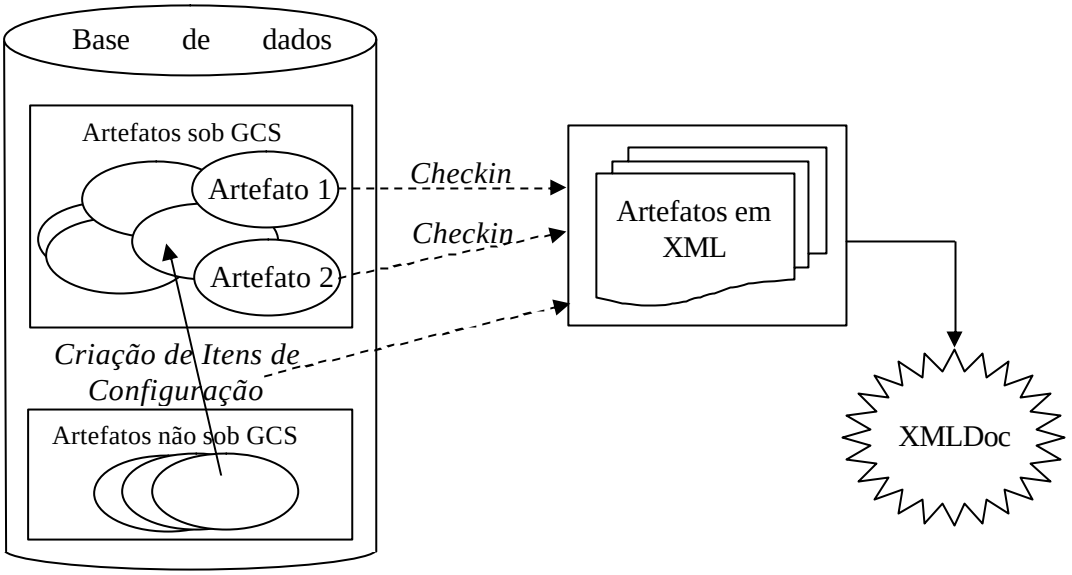


Figura 4 – A Infra-estrutura de GCS de ODE.

Os artefatos que não estiverem sob GCS podem ser alterados livremente. Porém, para que uma variação de um artefato sob GCS possa ser alterada, ela tem de ter uma alteração aprovada e o recurso humano acessando-a deve ser um dos responsáveis por tal alteração (restrição verificada durante o *check-out* dos itens). Somente nesses casos as ferramentas de elaboração dos respectivos artefatos são abertas, disponibilizando as funcionalidades para alteração dos mesmos. Nos demais casos, os usuários têm acesso apenas de leitura. Assim, não é aberta a correspondente ferramenta de apoio à elaboração do artefato, mas sim XMLDoc, a ferramenta de documentação, exibindo o arquivo XML correspondente à variação mais atual do artefato em questão (ou uma variação anterior, caso esta tenha sido especificada pelo usuário). Assim, alterações que por ventura estejam sendo feitas não são exibidas, uma vez que essas alterações estão sendo feitas apenas na base de dados (repositório central de ODE) e o que é exibido são as informações contidas nos arquivos XML.

Terminada uma alteração, no processo de *checkin*, é criado um arquivo XML correspondente à nova variação do item de configuração e, portanto, há arquivos XML para todas as variações dos itens de configuração, inclusive para a variação mais atual. Isso é importante, visto que algum desenvolvedor pode querer visualizar um artefato que esteja sob GCS e em alteração, não sendo ele um dos responsáveis por tal alteração. Neste caso, exibe-se, usando a ferramenta XMLDoc, o arquivo XML atual, ao invés das informações que constam no repositório central.

Com esse esquema, o impacto da introdução da GCS sobre as demais ferramentas é minimizado, uma vez que as ferramentas continuam a operar indistintamente sobre artefatos armazenados no repositório central de ODE, facilitando a integração. Passa a ser necessário, apenas, executar um método de controle de acesso à ferramenta, que verifica se o artefato está ou não sob GCS e, estando sob GCS, se houve uma alteração aprovada em que o usuário foi designado como um dos responsáveis pela alteração. Nas demais situações, o arquivo XML correspondente à variação mais atual do artefato é exibido usando XMLDoc,

não havendo, portanto, necessidade de maiores controles, uma vez que XMLDoc exibe as informações apenas para leitura.

Por fim, a memória organizacional, no que se refere aos artefatos de software, passa a não considerar todos os artefatos do repositório central de ODE, mas apenas aqueles que estão sob GCS. Isso é importante, já que artefatos em desenvolvimento, ainda não aprovados, muito provavelmente não estão maduros o suficiente para servirem de base para outros projetos. Além disso, como a partir da gerência de conhecimento, apenas são exibidos os artefatos, mas não alterados, o esquema usando arquivos XML e XMLDoc é bastante indicado.

No que se refere às funcionalidades implementadas na ferramenta de GCS, elas incluem serviços para apoiar as principais atividades do processo de GCS:

- **Colocar um Item sob Gerência de Configuração:** cria um novo item de configuração para o elemento que se está colocando sob GCS. Ao se criar um item de configuração para um artefato, é criada uma variação para ele e um arquivo XML. Além disso, deve-se informar, ainda, os recursos humanos que terão acesso (e o tipo de acesso), as sub-variações, no caso do artefato sendo colocado sob GCS ser composto de outros artefatos, e as variações dependentes, no caso do artefato sendo colocado sob GCS ter dependências com outros artefatos. A figura 5 ilustra essa situação.
- **Controlar Solicitação de Alteração**: permite acompanhar todo o processo de solicitação de alteração de um item de configuração do tipo artefato, incluindo o cadastro de uma solicitação de alteração e a aprovação (ou não) da solicitação.
- **Controlar Alteração**: trata da retirada para alteração (*check-out*) e do registro de uma alteração (*check-in*).
- **Relatar Estado de Configuração**: permite a um recurso humano autorizado verificar o estado de configuração de um artefato que esteja sob GCS.

5. Trabalhos Correlatos

Atualmente, há diversas ferramentas de GCS disponíveis [Estublier, 2000], implementando os principais serviços de apoio à GCS, como a ferramenta apresentada neste trabalho. Dois exemplos significativos são as ferramentas CVS (*Concurrent Version System*) [Caetano, 2004] e Rational ClearCase [Wahli et al., 2004], ambas adotando um esquema de GCS baseado em arquivos, em que há uma separação entre o repositório (um diretório que abriga todos os arquivos de um projeto sob GCS) e a área de trabalho do usuário (um diretório que o usuário utiliza para fazer suas alterações).

CVS é uma ferramenta isolada de GCS, de propósito geral, código aberto e multi-plataforma, amplamente utilizada. Uma opção inicialmente considerada foi integrá-la ao ambiente ODE. Contudo, essa opção foi descartada devido à sua abordagem centrada em arquivos e os potenciais problemas de integração decorrentes, sobretudo na dimensão de dados. Como os dados devem estar disponíveis a todas as ferramentas do ambiente, que podem usar informações parciais contidas em diversos artefatos, uma abordagem de

armazenamento em arquivos se mostra inadequada para um ADS. Por esse motivo, ODE adota como repositório de dados central um banco de dados relacional e, por conseguinte, a abordagem baseada em arquivos é insuficiente.

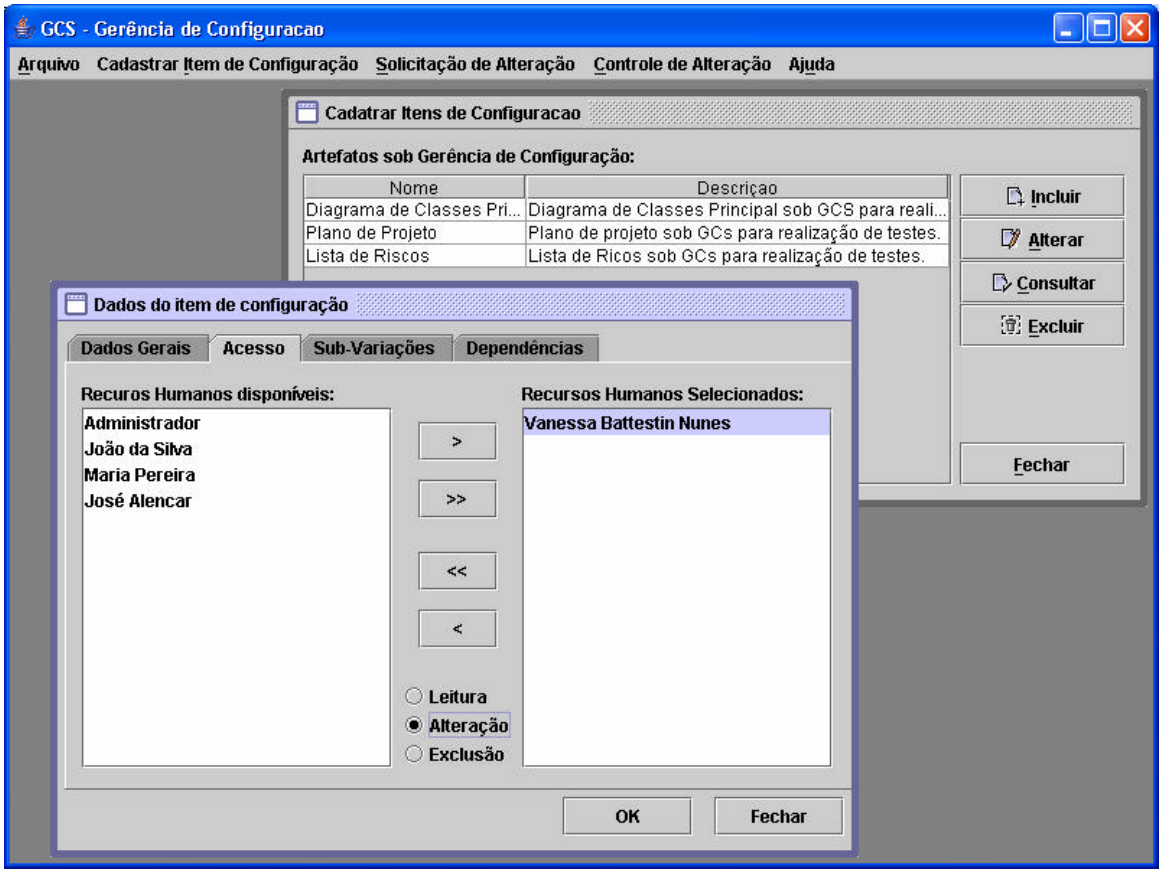


Figura 5 – Colocando um item sob GCS na ferramenta de GCS de ODE.

Rational ClearCase é uma ferramenta de GCS que permite controlar diversos "elementos" (requisitos, modelos, códigos-fonte, documentação, scripts de teste etc), construídos usando alguma das ferramentas da família de produtos IBM – Rational. Uma vez que essa família de produtos apóia um conjunto significativo de atividades do processo, conjuntamente, elas podem ser vistas como um ADS. Entretanto, como a persistência dos elementos é feita em arquivos, a abordagem adotada é a baseada em arquivos, considerada insuficiente para ODE, conforme mencionado anteriormente.

Por fim, para a Estação TABA foi desenvolvida a ferramenta GConf [Figueiredo et al., 2004], integrada aos ADSs TABA Orientados a Organização. GConf enfatiza a definição de um processo para a GCS e o registro e a consulta de conhecimento relacionado ao processo de GCS. Uma vez que a persistência na Estação TABA também é feita utilizando arquivos, a abordagem adotada não é suficiente para o ambiente ODE. Contudo, aspectos interessantes relacionados à gerência de conhecimento sobre o processo GCS contemplados por GConf poderão vir a ser considerados em ODE, já que este último também possui uma infra-estrutura de gerência de conhecimento associada.

6. Conclusões

Para que um ADS tenha suas ferramentas integradas, deve-se estabelecer uma infra-estrutura capaz de acomodar tais ferramentas, permitindo o compartilhamento de informações entre elas. A Gerência de Configuração de Software (GCS) aparece neste contexto com um papel principal: deve ser o centro de um ADS, controlando os artefatos produzidos e compartilhados pelas diversas ferramentas, estabelecendo e mantendo a integridade dos mesmos e garantindo que sejam apenas alterados pelos recursos humanos autorizados.

Este artigo apresentou a ferramenta de GCS de ODE, construída tomando por base uma ontologia de GCS. Desta forma, a ferramenta proposta está baseada em um modelo sólido, bem fundamentado, o que permitiu sua fácil integração no ambiente e, sobretudo com as demais ferramentas do mesmo.

A ontologia apresentada é uma importante contribuição, uma vez que pode servir de base para o entendimento compartilhado por uma comunidade de desenvolvimento de software, ou mesmo para apoiar a construção de outras ferramentas do gênero. Tal fato já se mostrou verdadeiro dentro do próprio contexto do projeto ODE, tendo sido a ontologia de GCS reutilizada por ocasião da construção de uma ontologia de requisitos de software [Nardi e Falbo, 2006], que está servindo de base para a construção de uma ferramenta de apoio à Engenharia de Requisitos.

Por fim, vale comentar alguns aspectos relacionados à implantação de ODE na empresa parceira relativos à GCS. O sistema proposto se mostra adequado, mas somente quando o artefato é produzido integralmente por uma ferramenta interna de ODE. Tendo em vista que ODE não possui ferramentas para cobrir todo o ciclo de vida e, mesmo quando há uma ferramenta disponível, muitas vezes, a organização já tem a cultura de adotar outra ferramenta (como no caso da ferramenta de apoio à modelagem usando UML), diversos artefatos são produzidos externamente a ODE e, portanto, não são tratados pela GCS do ambiente. Assim, no momento, está em curso um trabalho para tratar artefatos externamente produzidos e a gerência de configuração integrada dos mesmos.

Agradecimentos

Este trabalho foi realizado com o apoio do CNPq, entidade do Governo Brasileiro dedicada ao desenvolvimento científico e tecnológico, da FAPES, Fundação de Apoio à Ciência e Tecnologia do Espírito Santo, e da VixTeam Consultoria e Sistemas, empresa parceira que têm financiado o projeto e dado feedback de sua aplicação a casos reais.

Referências

Caetano, C. (2004) *CVS: Controle de Versões e Desenvolvimento Colaborativo de Software*. Editora Novatec.

Davies, J., Fensel, D., van Harmelen, F. (2003) *Towards The Semantic Web: Ontology-Driven Knowledge Management*, John Wiley & Sons Ltd.

Estublier, J. (2000), "Software Configuration Management: A Roadmap", In: Proc. of the Future of Software Engineering, ICSE'2000, Ireland.

Falbo, R. A. (1998) *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro.

Falbo, R. A. (2004) "Experiences in Using a Method for Building Domain Ontologies" Proc. of the 16th International Conference on Software Engineering and Knowledge Engineering, International Workshop on Ontology In Action, Banff, Canada.

Falbo, R. A., Natali, A. C. C., Mian, P.G., Bertollo, G., Ruy, F.B. (2003) "ODE: Ontology-based software Development Environment", In: Memórias de IX Congreso Argentino de Ciencias de la Computación, p. 1124-1135, La Plata, Argentina.

Figueiredo, S., Santos, G., Rocha, A.R.C. (2004) "Gerência de Configuração em Ambientes de Desenvolvimento de Software Orientados a Organização", In: Anais do III Simpósio Brasileiro de Qualidade de Software, Brasília.

Fuggetta, A. (2000), "Software Process: A Roadmap", In: Proc. of the Future of Software Engineering, ICSE'2000, Ireland.

Guarino, N. (1998) "Formal Ontology and Information Systems", In: Proceedings of the First Int. Conference on Formal Ontology in Information Systems, Trento, Italy.

Harrison, W., Ossher, H., Tarr, P. (2000), "Software Engineering Tools and Environments: A Roadmap", In: Proc. of the Future of Software Engineering, ICSE'2000, Ireland.

Mian, P.G. (2003) *ODEd: Uma Ferramenta de Apoio ao Desenvolvimento de Ontologias em um Ambiente de Desenvolvimento de Software*. Dissertação, Mestrado em Informática, UFES, Vitória.

Nardi, J.C., Falbo, R.A. (2006) "Uma Ontologia de Requisitos de Software", In: IX Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software – IDEAS´2006, La Plata, Argentina.

Natali, A.C.C., Falbo, R.A. (2003) "Gerência de Conhecimento em ODE", In: Anais do XVII Simpósio Brasileiro de Engenharia de Software, p. 270-285, Manaus, Brasil.

Nunes, V.B. (2005) *Integrando Gerência de Configuração de Software, Documentação e Gerência de Conhecimento em um Ambiente de Desenvolvimento de Software*. Dissertação, Mestrado em Informática, UFES, Vitória.

Nunes, V.B., Soares, A.O., Falbo, R.A. (2004) "Apoio à Documentação em um Ambiente de Desenvolvimento de Software, VII Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software, Arequipa, Peru.

Pfleeger, S.L. (2004) *Engenharia de Software: Teoria e Prática*, Prentice Hall, 2ª edição

Pressman, R. S. (2002), *Engenharia de Software*, Mc Graw Hill, 5a edição.

Sanches, R. (2001) "Gerência de Configuração", In: Qualidade e Produtividade em Software, 4a edição, Makron Books, Brasil.

SWEBOK (2001) *Guide to the Software Engineering Body of Knowledge*, IEEE Computer Society.

Wahli, U., Brown, J., Teinonen, M., Trulsson, L. (2004) “Software Configuration Management - A Clear Case for IBM Rational ClearCase and ClearQuest UCM”. International Technical Support Organization, ibm.com/redbooks.