Uso de Gerência de Conhecimento para Apoiar a Rastreabilidade e a Avaliação de Impacto de Alterações

Rodrigo Fernandes Calhau, Lucas de Oliveira Arantes, Ricardo de Almeida Falbo

Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, Brasil

{rodrigocalhau, lucasdeoliveira}@gmail.com, falbo@inf.ufes.br

Resumo. *O controle de alterações é uma parte essencial da Gerência de Configuração de Software. Quando uma alteração é feita, devem-se tomar os cuidados para que ela não comprometa a integridade de outros artefatos do projeto e, portanto, rastrear artefatos dependentes é fundamental para o controle de alterações. Este artigo apresenta uma abordagem baseada em gerência de conhecimento para aperfeiçoar a rastreabilidade de artefatos e apoiar a avaliação de impacto de alterações. Essa abordagem foi implementada no ambiente de desenvolvimento de software ODE.*

Abstract. *Change management is an essential part of the Software Configuration Management. When a change is done, developers should take care in order to jeopardize the integrity of other project artifacts. In this context, tracing dependent artifacts is fundamental. This paper presents a Knowledge Management based approach that aims to improve artifact dependency tracking and to support change impact evaluation. This approach was implemented in ODE, a software development environment.*

1. Introdução

Em qualquer projeto de software, há a geração de artefatos. Esses artefatos evidenciam a evolução do projeto e é muito importante registrar as modificações que ocorrem nos mesmos, de modo a se ter um histórico dessa evolução, o que pode ser feito por meio de um processo de Gerência de Configuração de Software (GCS). A GCS permite a uma organização de software manter um controle da evolução dos produtos de software, além de ajudar a cumprir metas de garantia da qualidade [Estublier 2000] [IEEE 2004]. Ela visa identificar a configuração de um sistema em diferentes pontos no tempo, com o propósito de sistematicamente controlar alterações e manter a integridade e a rastreabilidade da configuração ao longo do ciclo de vida do software [IEEE 2004].

Em uma organização que implementa a GCS, alterações nos principais artefatos devem ser controladas e podem, segundo a política da organização, necessitar de aprovação prévia para entrarem em vigor. Solicitações de alteração em artefatos devem ser analisadas pelo gerente de configuração, cabendo a ele negar ou permitir que uma alteração seja executada. Assim, o controle de alterações, ou gerência de mudanças, é uma parte importante da GCS.

O primeiro passo na gerência de mudanças consiste em determinar as alterações a serem feitas. Isso envolve, dentre outros, a submissão e o registro de uma solicitação de alteração, a avaliação do impacto da mesma e a aprovação, alteração ou rejeição da solicitação [IEEE 2004].

Quando uma solicitação de alteração é submetida por um usuário, ele pode ter dificuldade de identificar os artefatos a serem alterados, informando apenas que uma alteração no sistema é necessária. Neste caso, tipicamente um desenvolvedor deve analisar a solicitação e indicar quais artefatos potencialmente serão alterados. Quando um desenvolvedor faz a solicitação, normalmente ele já informa os artefatos a serem alterados. Entretanto, como no desenvolvimento de software há muita dependência entre artefatos produzidos por diferentes atividades do processo de software, é comum que uma alteração em um artefato tenha impactos em outros. Às vezes o desenvolvedor que analisou ou submeteu a alteração conhece essas dependências e já inclui na solicitação os artefatos dependentes. Outras vezes, não. Assim, cabe ao gerente de configuração, ao avaliar os impactos da alteração, analisar se o conjunto de artefatos enumerados para a alteração é adequado ou não, podendo incluir ou excluir artefatos nessa lista. Neste contexto, manter a rastreabilidade bidirecional entre os principais artefatos é fundamental para apoiar a tomada de decisão [Antoniol et al. 2002].

Para se manter a rastreabilidade entre artefatos, é interessante que sejam definidas as dependências entre eles. Apesar de ser uma abordagem valiosa para o bom funcionamento da GCS, muitas vezes esse mapa de dependências não é formalmente estabelecido. A não observância das dependências entre artefatos pode gerar problemas, tal como a alteração em um artefato vir a causar inconsistências em seus dependentes.

Para tentar contornar esse problema, pode-se adotar uma política organizacional de registro de dependências entre tipos de artefatos, sinalizando que uma alteração em um artefato de certo tipo pode causar impacto em artefatos de outros tipos, dele dependentes. Manter esse mapa de dependências entre os tipos de artefatos ao alcance dos membros da organização permite apoiar a análise de impacto de alterações, servindo de base para rastrear os possíveis artefatos que devem ser levados em conta em uma dada alteração. Além disso, é interessante que essa facilidade seja provida de forma automatizada, provendo sugestões para o solicitador de uma alteração ou para o gerente de configuração de quais outros artefatos devem ser considerados em uma alteração, dado um conjunto inicial informado.

Entretanto, apenas a sugestão com base em um mapa de dependências organizacional pode não ser suficiente. Mesmo quando uma organização define esse mapa, para artefatos ou projetos específicos, ele pode ser aperfeiçoado se observadas experiências anteriores, sobretudo em situações similares. Além disso, com o passar do tempo, o mapa de dependências tende a se tornar desatualizado, abrindo espaço para a ocorrência de erros na análise de impacto de alterações. Nesse cenário é importante observar, não só um mapa de dependências organizacional, mas também qual é a tendência de evolução desse mapa, analisando os artefatos envolvidos em alterações registradas nos projetos da organização.

Este artigo discute o uso de Gerência de Conhecimento para apoiar a rastreabilidade e a avaliação de impacto de alterações em artefatos de software mantidos sob gerência de configuração, apresentando formas de apoio automatizado relacionadas à identificação de dependências. A mais simples delas permite a elaboração do mapa de dependências organizacional entre tipos de artefatos e.o uso de dados de alterações realizadas em projetos da organização para a atualização contínua do mesmo. Para complementar a informação provida pelo mapa de dependências no contexto de um

projeto específico, sugestões de dependências entre artefatos podem ser adicionadas tomando por base análises de alterações em projetos similares ou levando-se em consideração alterações anteriormente realizadas no próprio projeto. Funcionalidades para apoiar cada uma dessas abordagens foram desenvolvidas no ambiente ODE (*Ontology-based software Development Environment*) [Falbo et al. 2003], utilizando seu sistema de gerência de configuração e sua infra-estrutura de gerência de conhecimento.

A organização deste artigo é a seguinte: a seção 2 discute o referencial teórico envolvido, o que inclui GCS, Rastreabilidade, Gerência de Conhecimento e o ambiente ODE, no qual este trabalho foi materializado; a seção 3 apresenta a nova versão do sistema de GCS de ODE, procurando destacar aspectos relacionados à rastreabilidade de dependências entre artefatos e seu uso no controle de alterações; a seção 4 discute o uso de gerência de conhecimento para apoiar a rastreabilidade de artefatos e a análise de impacto de alterações; finalmente, a seção 5 discute brevemente trabalhos correlatos e a seção 6 apresenta as conclusões deste trabalho.

2. Gerência de Configuração de Software, Rastreabilidade e Gerência de Conhecimento

Segundo Estublier (2000), a Gerência de Configuração de Software (GCS) é a disciplina de Engenharia de Software que permite manter sob controle os produtos de software que evoluem, contribuindo, assim, para a satisfação de restrições de qualidade e tempo. A importância da GCS é reconhecida pelos vários modelos e normas de qualidade de processo, sendo considerada uma das mais bem sucedidas tecnologias da Engenharia de Software [Estublier 2000]. O modelo CMMI [SEI, 2006], por exemplo, define a GCS como uma área de processo do nível 2 de maturidade.

De maneira geral, o processo de GCS envolve atividades relacionadas a [IEEE 2004] [ISO/IEC 2008]: gerência do processo de GCS, identificação da configuração de software, controle de alterações, registro e apresentação da situação dos itens e das solicitações de alteração, garantia da consistência dos itens alterados e controle do armazenamento, manipulação e distribuição de itens.

Na identificação da configuração, devem ser identificados 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. Os relacionamentos estruturais entre itens e suas partes constituintes e as dependências entre itens afetam outras atividades da GCS, sobretudo o controle de alterações, pois permitem identificar de maneira eficaz os itens afetados em decorrência de uma alteração [IEEE 2004] [Sanches 2001].

Alterações realizadas sem controle podem levar rapidamente ao caos. Assim, a GCS deve instituir um sub-processo de Controle de Alterações ou Gerência de Mudanças, combinando procedimentos humanos e ferramentas automatizadas. Nesse processo, quando um usuário detecta a necessidade de alteração em algum item de configuração, ele deve submeter uma solicitação de alteração. Essa solicitação de alteração deve ser analisada por um desenvolvedor para que ele avalie, dentre outros, o mérito técnico, potenciais efeitos colaterais e o impacto global sobre outros itens de configuração e funções do sistema. Os resultados dessa avaliação são apresentados como um relatório de alteração que deve ser analisado pelo gerente de configuração, um

papel organizacional que pode ser desempenhado por uma pessoa ou um grupo, que toma a decisão final sobre o status e a prioridade da alteração. Se aprovada a solicitação, passa-se à execução da alteração, quando os itens a serem alterados são disponibilizados para os responsáveis pela alteração (registro de saída ou *checkout*), que realizam o trabalho necessário. Uma vez alterados, os itens devem ser submetidos a atividades de garantia da qualidade e, se aprovados, registrados (registro de entrada ou *checkin*), quando os mecanismos apropriados de controle de versão são aplicados [Pressman 2006].

Quando o desenvolvedor analisa uma solicitação de alteração, ele deve, dentre outros, avaliar o impacto da alteração sobre outros itens de configuração, o que pressupõe uma análise das dependências entre os diversos artefatos produzidos no projeto. De maneira análoga, quando o gerente de configuração revisa a lista de artefatos a serem potencialmente alterados, adicionando ou removendo itens, é muito importante que os relacionamentos entre os itens de configuração estejam registrados. A habilidade de manter o rastro desses relacionamentos é crucial para a integridade dos produtos de trabalho [Pressman, 2006]. Contudo, manter a rastreabilidade bidirecional de artefatos e suas versões não é uma tarefa trivial e, portanto, devem-se prover ferramentas e mecanismos que ajudem a manter a rastreabilidade e, por conseguinte, que apóiem desenvolvedores e gerentes de configuração na difícil tarefa de avaliar os impactos de uma alteração.

Manter políticas organizacionais descrevendo mapas de dependência típicos entre tipos de artefatos é um bom ponto de partida para apoiar a manutenção da rastreabilidade. Contudo, esse mecanismo, aplicado isoladamente, pode não ser suficiente. Dependendo das características de diferentes projetos, pode ser necessário incluir ou alterar relacionamentos de dependência. Seja, por exemplo, a classe de projetos voltados para o desenvolvimento de aplicações *Web*. Neste caso, a organização pode preconizar o desenvolvimento de modelos de navegação, artefato não tipicamente produzido em projetos de sistemas de informação convencionais para a plataforma *desktop*. De fato, seria necessário manter mapas de dependências para cada classe de projetos diferente. Mas mesmo assim, há de se considerar que o desenvolvimento de software é uma atividade dinâmica e que, com o passar do tempo, os mapas de dependências podem se tornar desatualizados. Assim, é necessário melhorá-los continuamente. Para apoiar a solução desses problemas, abordagens de Gerência de Conhecimento podem ser aplicadas.

A Gerência de Conhecimento (GC) visa à administração sistemática e ativa dos recursos de conhecimento de uma organização, utilizando tecnologia adequada, com vistas ao compartilhamento e aprendizado organizacional [Lima 2004]. Ela fornece as bases para a construção de repositórios de conhecimento organizacionais, bem como para o provimento de serviços de apoio à captura, disseminação, uso e evolução do conhecimento, de modo que o conhecimento gerado na organização seja armazenado de uma forma que facilite a sua recuperação e utilização. Um aspecto central para a GC é a reutilização de experiências anteriores, preferencialmente em situações similares.

No contexto do desenvolvimento de software, a GC pode facilitar a aquisição de conhecimento sobre novas tecnologias, o acesso ao conhecimento de domínio, a troca de conhecimento acerca de práticas e políticas locais, etc [Rus e Lindvall. 2002]. Mais

especificamente, no contexto discutido anteriormente, o uso de experiências anteriores em situações similares pode ser uma abordagem interessante para complementar os mapas de dependências entre artefatos, ou mesmo para a melhoria desses mapas, provendo indicação de relacionamentos não explicitamente declarados e apresentando pró-ativamente para desenvolvedores e gerentes de configuração artefatos que podem ser impactados em uma alteração.

Para ser efetivamente utilizado, um sistema de GC deve estar integrado ao ambiente de trabalho existente. No caso do desenvolvimento de software, esse ambiente de trabalho pode ser um Ambiente de Desenvolvimento de Software e, portanto, é importante que as facilidades de GC estejam disponíveis no ambiente como um todo, e mais especificamente, dado o foco deste trabalho, no sistema de gerência de configuração. Assim, neste trabalho foram providas funcionalidades baseadas em GC no sistema de gerência de configuração do ambiente ODE, denominado GCS-ODE.

ODE (*Ontology-based software Development Environment*) [Falbo et al., 2003] é um Ambiente de Desenvolvimento de Software Centrado em Processo, desenvolvido tomando por base ontologias do domínio de Engenharia de Software. Sua arquitetura conceitual é organizada em três níveis com o intuito de amarrar semanticamente os seus objetos por meio de anotações de meta-dados baseados em ontologias [Falbo et al. 2005]. O Nível Ontológico é responsável pela descrição das ontologias em si. O Nível de Conhecimento abriga as classes que descrevem o conhecimento sobre tipos de objetos, os quais devem ser definidos como conceitos no Nível Ontológico. Já o Nível da Aplicação define as classes responsáveis por implementar as aplicações no contexto do ambiente e que tratam das classes de instâncias dos tipos do nível de conhecimento.

ODE possui diversas ferramentas integradas e provê para suas ferramentas algumas infra-estruturas que podem ser utilizadas no apoio a diferentes atividades, dentre elas uma infra-estrutura de gerência de conhecimento [Falbo et al., 2004]. A infra-estrutura de gerência de conhecimento é suportada por uma infra-estrutura de caracterização de itens de software [Carvalho et al., 2006] que provê uma abordagem flexível para caracterizar itens de software e calcular similaridade entre eles, fazendo uso de técnicas de raciocínio baseado em casos, para apoiar o uso de experiências anteriores similares. Em relação às ferramentas de ODE, a ferramenta foco deste trabalho é o sistema de gerência de configuração de software, apresentado a seguir.

3. GCS-ODE: o Sistema de Gerência de Configuração de ODE

A primeira versão GCS-ODE, apresentada em [Nunes e Falbo, 2006], foi desenvolvida tomando por base uma ontologia de gerência de configuração de software, procurando oferecer serviços básicos de controle de versão e de controle de alteração. Nessa versão, o repositório da GCS era composto do repositório central do ambiente e de arquivos XML, sendo as funcionalidades providas para o apoio ao controle de versões bastante limitadas, quando comparadas a de outros, sistemas como o CVS (*Concurrent Version System*) [Caetano 2004] ou o Subversion (SVN) [Nagel 2005].

De fato, constatou-se que dificilmente seria possível prover as funcionalidades oferecidas por diversos sistemas de controle de versão, devido às limitações de recursos do Projeto ODE. Assim, em uma nova versão, decidiu-se integrar GCS-ODE ao SVN. A construção dessa nova versão do sistema de GCS foi feita com base em uma evolução

da ontologia de GCS [Arantes et al. 2007] e diversas funcionalidades foram remodeladas e reconstruídas. A Figura 1 mostra um excerto dessa ontologia.

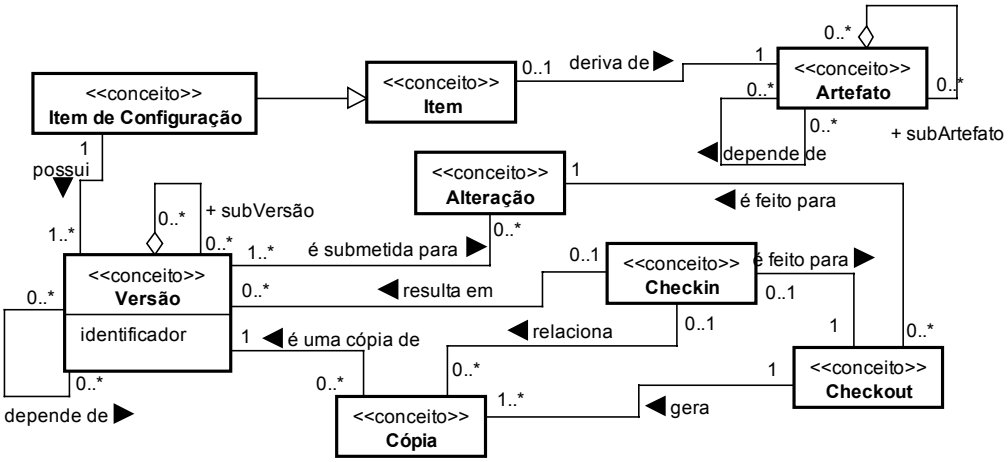

Figura 1 – Excerto da Ontologia de Gerência de Configuração de Software.

Como mostra a Figura 1, itens de configuração derivam de artefatos e possuem versões, as quais podem ser submetidas a alterações. Uma retirada (*checkout*) é feita para uma alteração, gerando cópias das versões em alteração. Quando concluído o trabalho, um *checkin* é feito, relacionando as cópias que dão origem a novas versões.

Essa ontologia captura, ainda, dois aspectos fundamentais para a rastreabilidade de artefatos: relacionamentos estruturais entre artefatos e suas partes constituintes (sub-artefatos) e as dependências entre artefatos. Essa estrutura é replicada no nível de versões, permitindo indicar as dependências entre versões específicas.

Na atual versão, GCS-ODE fornece apoio a algumas atividades do controle de versão e do controle de alterações. Para o controle de versão, integrou-se o SVN a ODE e, assim, grande parte das funcionalidades providas pelo SVN estão disponíveis em GCS-ODE. Contudo, aspectos dessa integração não são foco deste trabalho e, portanto, não são aqui discutidos.

No que se refere ao controle de alterações, são oferecidas diversas funcionalidades, tomando por base, agora, a nova versão da ontologia de GCS. Uma alteração em ODE envolve as seguintes etapas: seleção dos artefatos que estão sob GCS (itens de configuração), retirada dos mesmos (*checkout*) e criação de uma cópia de trabalho, realização da alteração na cópia de trabalho gerada e, por último, o registro das alterações realizadas (*checkin*).

Em cada alteração deve-se indicar um artefato que é tido como o artefato principal da alteração e, a partir dessa informação, podem-se selecionar outros artefatos que serão possivelmente impactados pela alteração do artefato principal. Dessa forma a seleção dos artefatos pode ser dividida em duas partes: seleção do artefato principal, aquele que motiva a realização da alteração, e a seleção dos artefatos dependentes, que poderão ser impactados com a alteração do artefato principal.

Conforme mencionado anteriormente, o mapeamento de ontologias no ambiente ODE se dá em dois níveis: o nível de conhecimento que basicamente trata de tipos e o nível de aplicação que trata das classes de instâncias dos tipos do nível de

conhecimento. O conceito *Artefato* da ontologia está presente tanto no nível de conhecimento quanto no nível de aplicação. No nível de conhecimento, a classe *KArtefato* representa tipos de artefatos. No nível de aplicação, a classe *Artefato* representa artefatos concretos, pertencentes a um projeto. Todo artefato no nível de aplicação possui um tipo definido no nível de conhecimento, como ilustra a Figura 2. Dessa forma, em ODE, as dependências entre artefatos podem ser representadas nesses dois níveis.

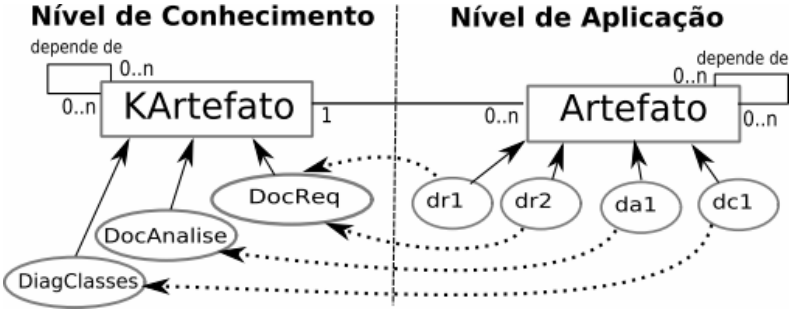

Figura 2 – Representação de Artefatos nos Níveis de Conhecimento e Aplicação da Arquitetura Conceitual de ODE.

A dependência entre dois tipos de artefatos indica que uma alteração em um artefato de um tipo pode causar impacto em algum artefato do outro tipo. Ou seja, se um tipo *t1* depende do tipo *t2*, então uma alteração de um artefato *a2* do tipo *t2* pode causar impacto em algum artefato *a1* do tipo *t1*. Com a representação de dependências entre tipos, é possível representar um mapa organizacional de dependências entre artefatos.

A Figura 3 ilustra um exemplo de política organizacional de dependência entre artefatos, válida para todos os projetos de uma organização. Nessa figura, assim como no restante deste trabalho, não fazemos distinção entre relacionamentos estruturais de artefatos e suas partes (sub-artefatos) e relacionamentos de dependência entre artefatos, passando a tratá-los uniformemente como dependência. Isso decorre do axioma da ontologia de GCS que aponta que, se um artefato *a2* é parte de um artefato *a1*, então *a1* depende de *a2*: $(\forall\ a1,a2)\ (parteDe\ (a2, a1)) \rightarrow dependeDe\ (a1,a2)$.

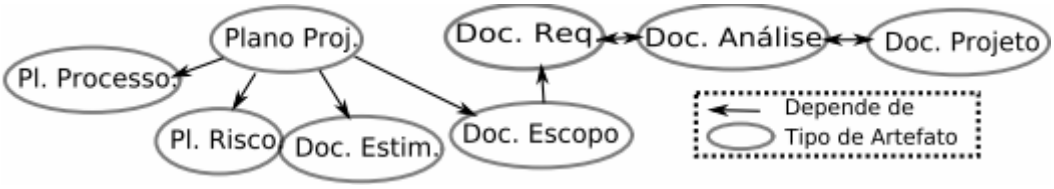

Figura 3 – Exemplo de Política Organizacional de Dependência entre Artefatos.

Como mostra o exemplo da Figura 3, definiu-se em uma organização que planos de projeto tipicamente dependem de suas partes constituintes: planos de processo, planos de risco, documentos de estimativas e documentos de escopo. Já documentos de escopo são dependentes de documentos de requisitos. Documentos de requisitos e de análise são interdependentes, assim como documentos de análise e projeto.

A dependência entre dois artefatos concretos de um projeto, no nível de aplicação de ODE, indica que uma alteração em um deles pode causar impacto no outro. Cada alteração em uma versão de um artefato dá origem a uma nova versão. As versões

também possuem dependências, mas entre versões, automaticamente estabelecidas no momento da criação da versão, com base nas dependências entre artefatos.

A Figura 4 ilustra um exemplo de dependências entre artefatos pertencentes a um projeto, baseadas na política organizacional ilustrada na Figura 3. Suponha que em tal projeto desenvolve-se um sistema dividido em dois subsistemas e que, por isso, há documentos distintos de requisitos, de análise e de projeto para cada um dos subsistemas. Na definição das dependências de um dado artefato, são sugeridos outros artefatos do mesmo projeto, tomando por base as dependências entre o tipo do artefato em questão e os tipos dos outros artefatos, conforme definido no mapa de dependências da organização. Por exemplo, na definição das dependências do artefato *Doc.Análise1*, são sugeridos como artefatos dependentes *Doc.Req.1, Doc.Req.2, Doc.Proj.1* e *Doc.Proj.2*, uma vez que *Doc.Análise1* é do tipo *Documento de Análise* (*Doc.Análise)*, que depende dos tipos *Documento de Requisitos* (*Doc.Req.)* e *Documento de Projeto* (*Doc.Proj.)*. No exemplo da Figura 4, apenas *Doc.Req.1* e *Doc.Proj.1* foram selecionados. Deve-se realçar que é possível selecionar artefatos que não foram sugeridos, bem como não selecionar nenhum dos artefatos sugeridos.

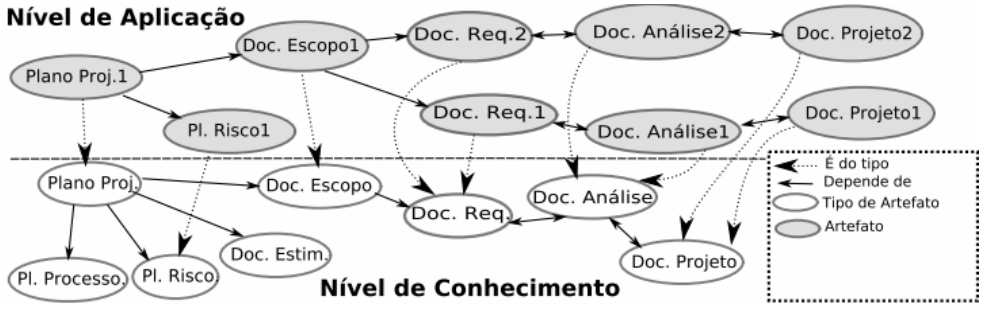

Figura 4 – Exemplo de Dependências entre Artefatos de um Projeto.

Da mesma forma que as dependências entre tipos de artefatos são usadas para prover sugestões durante a definição das dependências entre artefatos, na seleção de artefatos para a realização de uma alteração, são feitas sugestões de artefatos que podem ser impactados pela alteração do artefato principal dessa alteração, tomando por base as dependências entre artefatos. Seja o exemplo da Figura 4. Em uma alteração do artefato *Doc.Req.2*, por exemplo, seriam sugeridos os artefatos *Doc.Escopo1* e *Doc.Análise2*, que são justamente os artefatos que dependem dele.

A Figura 5 ilustra um cenário com as etapas de uma alteração em artefatos do projeto do exemplo anterior. Na parte (a) é ilustrada a seleção do artefato principal *Doc.Análise.1* e a sugestão dos artefatos que podem ser impactados por ele, de acordo com as dependências apresentadas na Figura 4. Foram sugeridos os artefatos *Doc.Req.1* e *Doc.Projeto.1*, que são dependentes de *Doc.Análise.1*. Porém *Doc.Req.1* também possui artefatos dependentes e, portanto, foram sugeridos, também, os artefatos *Doc.Escopo1,* que depende de *Doc.Req.1,* e *Plano Proj.1* que depende de *Doc.Escopo1*.

A parte (b) ilustra a seleção dos artefatos para alteração que o desenvolvedor julga que podem ser impactados. Nota-se que é possível selecionar um artefato que não foi sugerido (*Doc.Análise2* no caso), assim como é possível não selecionar itens que foram indicados como potencialmente impactados (no exemplo, *Doc.Escopo1*, *Plano Proj.1* e *Doc.Projeto.1*).

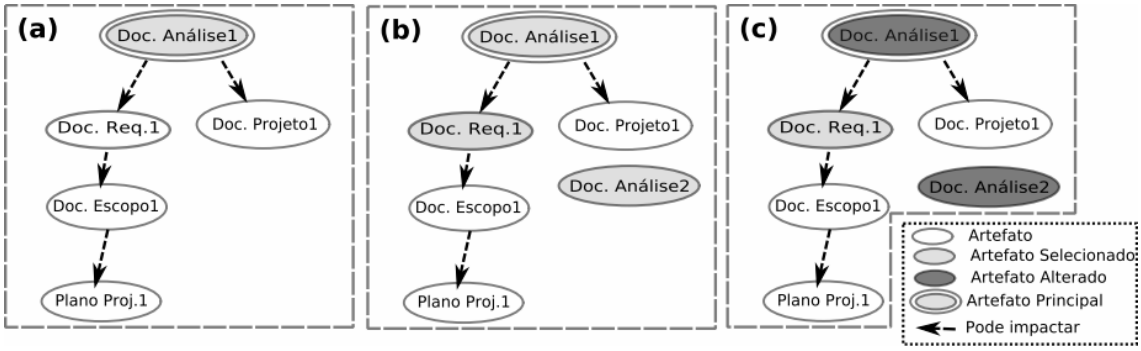

Figura 5 – Alterações em ODE: (a) Seleção do artefato principal da alteração e sugestão de artefatos que podem ser impactados; (b) Conjunto de artefatos selecionados para alteração; (c) Subconjunto dos artefatos selecionados que foram efetivamente alterados.

Por último, a parte (c) apresenta os artefatos que foram efetivamente alterados no registro da alteração (*Doc.Análise1* e *Doc.Análise2*). Dessa forma, cada alteração possui um artefato principal (no exemplo, *Doc.Análise1*), um conjunto de artefatos selecionados para a alteração (*Doc.Análise1*, *Doc.Req.1* e *Doc.Análise2*) e um conjunto de artefatos efetivamente alterados (*Doc.Análise1* e *Doc.Análise2*). O conjunto de artefatos efetivamente alterados em uma alteração é obrigatoriamente um subconjunto do conjunto de artefatos selecionados nessa alteração.

4. Apoio de Gerência de Conhecimento no Controle de Alterações de ODE

As sugestões baseadas nas dependências entre tipos de artefatos e entre artefatos, discutidas na seção anterior, são um importante meio para se manter a integridade dos artefatos de um projeto, à medida que alterações vão ocorrendo. Porém tais tipos de sugestões ainda possuem limitações. Não é oferecido, por exemplo, nenhum apoio à atualização do mapa de dependências organizacional. Ela fica totalmente por conta do conhecimento do responsável, tipicamente, o gerente de configuração. Além disso, deve-se considerar que projetos diferentes possuem características diferentes. Dessa forma, uma única política organizacional de dependências pode ser insuficiente.

Neste contexto, o uso de Gerência de Conhecimento na análise do histórico das alterações ocorridas em uma organização pode fornecer informações bastante importantes sobre as reais dependências entre artefatos da organização. Esse histórico pode, por exemplo, dar pistas indicando a possível existência de uma dependência entre artefatos, não capturada no mapa organizacional.

Neste trabalho, o apoio de gerência de conhecimento oferecido no contexto da GCS se dá por meio de sugestões baseadas no histórico de alterações. Essas sugestões ocorrem em três ocasiões, como mostra a Figura 6: (a) na atualização do mapa de dependências entre tipos de artefatos; (b) na definição do mapa de dependências entre artefatos de um projeto; ou (c) na seleção de artefatos para alteração. A seguir, cada um desses tipos de sugestão baseada no histórico de alterações é descrito com mais detalhes.

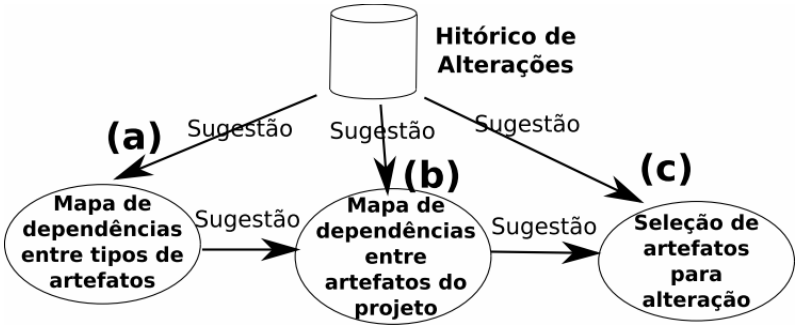

Figura 6 – Tipos de Sugestões baseadas no Histórico de Alterações.

(a) Sugestão de Dependências entre Tipos de Artefatos

Para prover sugestões de dependências entre tipos de artefatos durante a atualização do mapa de dependências organizacional, são analisadas as alterações ocorridas nos vários projetos da organização. Para cada tipo de artefato, são sugeridas dependências com base na análise dos artefatos principais das alterações e dos conjuntos de artefatos efetivamente impactados por elas.

Seja, por exemplo, o tipo de artefato *Documento de Requisitos (Doc.Req.*) do mapa de dependências organizacional apresentado na Figura 3. São obtidas todas as alterações cujo artefato principal é do tipo *Documento de Requisitos.* A partir delas, são analisados os tipos de cada um dos artefatos pertencentes aos conjuntos de artefatos efetivamente impactados, com o intuito de saber o conjunto de tipos impactados e a freqüência desse impacto. Suponha, por exemplo, que em 45% das alterações cujo artefato principal foi do tipo *Documento de Requisitos* um artefato do tipo *Documento de Estimativas (Doc.Estim.)* foi impactado. Assim, na definição das dependências do tipo de artefato *Documento de Estimativas* será sugerida uma dependência com o tipo *Documento de Requisitos*, informando a porcentagem de impacto calculada.

As porcentagens de impactos de tipos de artefatos cujas dependências já estão no mapa da organização também são informadas, facilitando, assim, não só a adição de dependências, mas também a remoção delas.

A Figura 7 ilustra a definição de dependências para o exemplo acima, considerando *Documento de Estimativas* o tipo de artefato base, isto é, aquele para o qual se deseja definir os tipos de artefatos dos quais esse tipo depende. Nota-se na tabela à direita que não há nenhuma dependência definida para esse tipo de artefato no mapa organizacional. Na tabela da esquerda são apresentados os tipos de artefatos indicados com base no histórico de alterações, com as respectivas porcentagens de impacto, podendo ser essa lista filtrada pela porcentagem de impacto. No exemplo, está sendo apresentado apenas o tipo de artefato *Documento de Requisitos*, pois é o único que leva a alterações em *Documentos de Estimativa* em mais de 40% das vezes em que é alterado.

(b) Sugestão de Dependências entre Artefatos

Conforme discutido na seção 3, a dependência entre artefatos é feita principalmente a partir do mapa de dependências da organização. Contudo, outras duas formas de sugestão são providas: levando-se em conta o histórico de alterações de projetos similares ou considerando apenas o histórico de alterações do próprio projeto.

Figura 7 –Dependências entre Tipos de Artefatos.

A sugestão baseada em projetos similares é análoga à sugestão de tipos de artefatos. A diferença é que, neste caso, considera-se apenas o histórico de alterações ocorridas em projetos similares ao projeto corrente. Assim, a sugestão de dependências baseada em projetos similares é feita ainda entre tipos de artefatos, sendo aplicada, porém, nos artefatos do projeto, durante a definição de suas dependências.

Seja o projeto de exemplo da Figura 4. Suponha que para a classe de projetos desse projeto, o histórico de alterações da organização aponte que em 70% das vezes em que um artefato do tipo *Documento de Requisitos* é alterado, um artefato do tipo *Plano de Riscos (Pl.Risco)* também o é. Assim, na definição das dependências do artefato *Pl.Risco1* são sugeridos os artefatos do projeto que são do tipo *Documento de Requisitos*, ou seja, *Doc.Req.1* e *Doc.Req.*2. Vale ressaltar que, se a definição de dependências fosse para um outro artefato *Pl.Risco2* também do tipo *Plano de Riscos*, então a sugestão baseada nos projetos similares seria a mesma, visto que ela é feita com base nos tipos de artefatos.

Esse tipo de sugestão é importante na identificação de dependências que ocorrem em classes de projetos específicas. Assim, ela é uma forma de tornar o mapa de dependências organizacional mais sensível às características de classes de projetos. Neste contexto, a infra-estrutura de caracterização de projetos de ODE é de suma importância, pois é ela que provê o cálculo de similaridade entre projetos.

Já a sugestão de dependências entre artefatos baseada apenas no histórico do projeto corrente não leva em consideração as alterações ocorridas em outros projetos. Essa sugestão também não leva em consideração os tipos dos artefatos, como a sugestão baseada em projetos similares, mas apenas as alterações ocorridas nos artefatos do projeto em questão.

Seja novamente o exemplo da Figura 4, considerando os artefatos *Doc.Análise1* e *Doc.Análise2*. Ao se fazer uma análise das alterações em que o artefato principal foi *Doc.Análise1*, pode-se chegar à conclusão que em 60% das vezes o artefato *Doc.Análise2* foi também alterado. Por outro lado, ao se analisar as alterações em que o artefato principal foi *Doc.Análise2*, pode-se chegar à conclusão de que em 40% delas o artefato *Doc.Análise1* foi também alterado. Isso pode indicar uma dependência mútua entre esses dois artefatos.

Vale ressaltar que a porcentagem de impacto de alteração em artefatos previamente considerados dependentes também é apresentada, ajudando, assim, na remoção de dependências entre artefatos.

Esse tipo de sugestão é uma forma alternativa à baseada em projetos similares, visto que ele faz uma análise de impacto em termos dos artefatos de um mesmo projeto, considerando, assim, as particularidades desse projeto. Ambos os tipos de sugestão são, portanto, complementares, pois há momentos em um projeto, tipicamente no início, que seu histórico de alterações ainda não está rico o suficiente para sugerir dependências de maneira mais precisa. A Figura 8 ilustra os exemplos das sugestões de dependências baseadas no histórico de alterações discutidos anteriormente.

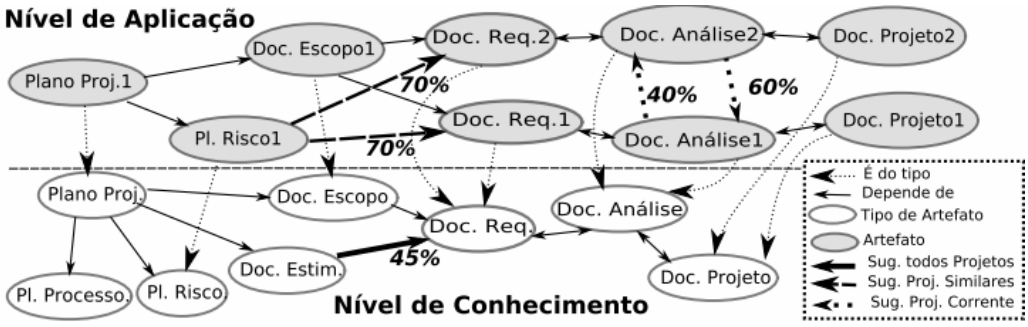

Figura 8 – Sugestões de Dependências entre Tipos de Artefatos e entre Artefatos baseadas no Histórico de Alterações.

(c) Sugestão de Artefatos para Alteração

A sugestão de artefatos para alteração é feita com base nas dependências entre artefatos estabelecidas no caso anterior (item (b)). Para cada artefato dependente do artefato principal da alteração, é apresentada uma porcentagem de impacto, calculada com base no histórico do próprio projeto. A forma como essa porcentagem é calculada é a mesma forma como são calculadas as porcentagens apresentadas nas sugestões de dependência entre artefatos com base no histórico do próprio projeto.

A Figura 9 ilustra um exemplo de sugestão de artefatos para alteração em uma solicitação de alteração cujo artefato principal é *Doc.Análise1*. O histórico de alterações do projeto aponta, por exemplo, que em 35% das vezes em que o artefato *Doc.Análise1* foi alterado, o artefato *Doc.Req.1* também o foi; que em 20% das vezes em que *Doc.Análise.1* foi alterado, o artefato *Doc.Escopo1* também foi alterado; que em 15% das vezes em que *Doc.Análise.1* foi alterado, *PlanoProj.1* também o foi; e finalmente que em 45% das vezes em que *Doc.Análise.1* foi alterado, *Doc.Projeto1* também o foi.

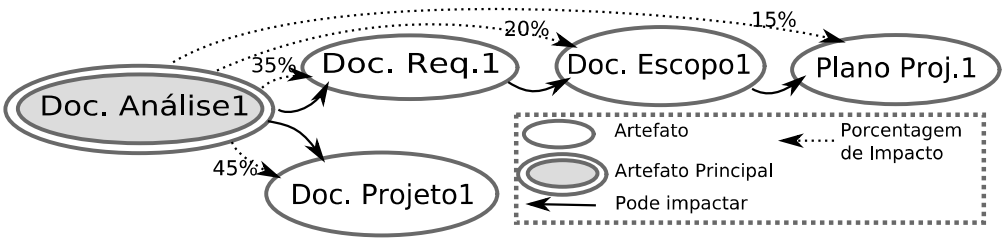

Figura 9 - Sugestões de Artefatos para Alteração.

Esse mesmo exemplo é mostrado na Figura 10. Após o desenvolvedor selecionar a ramificação (*trunk*) e o artefato principal da alteração (*Doc. Análise 1*), ele deve selecionar aqueles artefatos que julga serem potencialmente impactados pela alteração do artefato principal. Há duas opções: listar todos os artefatos contidos na ramificação selecionada ou listar apenas os artefatos dependentes do artefato principal da alteração, conforme a definição de dependências no contexto do projeto. Em ambos os casos, são apresentados os artefatos e as respectivas porcentagens de impacto.

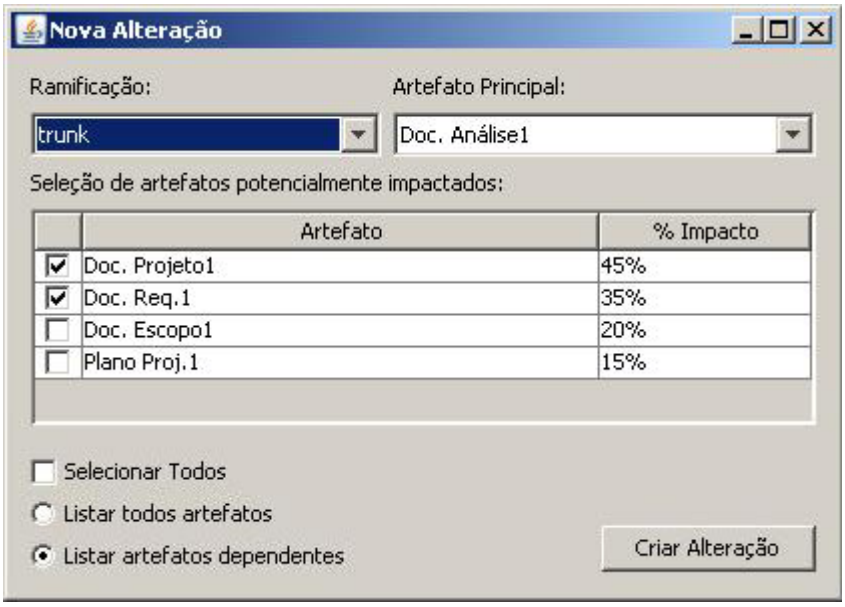

Figura 10 – Sugestão de Artefatos para alteração em ODE.

Apesar da sugestão de artefatos para alteração parecer ser a mais importante, vale destacar que as demais sugestões são igualmente importantes, visto que elas contribuem indiretamente com a seleção dos artefatos para alteração. Como o mapa de dependências entre tipos de artefatos é a base para a sugestão de dependências entre artefatos, e este, por sua vez, é a base para a sugestão de artefatos para alteração, quanto melhores forem os mapas de dependência de tipos de artefatos e de artefatos, melhor será a sugestão de artefatos para alteração. Na realidade, a sugestão baseada no histórico de alterações que ocorre na atualização dos mapas de dependências possui justamente esse papel: o de melhorar o mapa organizacional, a fim de melhorar a seleção dos artefatos para uma alteração. Assim, todos os tipos de sugestão discutidos são importantes. Eles estão relacionados e contribuem, direta ou indiretamente, para se manter a integridade dos artefatos de um projeto.

5. Trabalhos Correlatos

Comparando GCS-ODE com alguns sistemas disponíveis e amplamente usados, dentre eles o CVS (*Concurrent Version System*) [Caetano 2004] e o *Subversion* (SVN) [Nagel 2005], pode-se perceber que a atual versão de GCS-ODE provê um bom nível de controle de versão, uma vez que, de fato, utiliza as funcionalidades do SVN para tal, e um controle de alterações superior. Nos dois sistemas citados, para que um uma alteração seja realizada, basta que os itens estejam liberados para sofrer alterações. Em GCS-ODE, por sua vez, antes de realizar uma alteração, o desenvolvedor deve submeter uma solicitação de alteração, que tem de ser aprovada pelo gerente de configuração. O mesmo ocorre com as alterações implementadas, que só dão origem a novas versões se forem aprovadas após serem analisadas pelo gerente de configuração.

Sob essa ótica, ainda, há outras ferramentas que provêem o mesmo nível de funcionalidades de apoio ao controle de alterações ou até superiores, tais como *GConf* [Figueiredo et al. 2004], a ferramenta de GCS da Estação TABA, e Odyssey-SCM, a infra-estrutura de GCS do ambiente Odyssey [Murta 2006]. Vale destacar que, assim como GCS-ODE, esses dois sistemas estão integrados a Ambientes de Desenvolvimento de Software Centrados em Processo e se beneficiarem, portanto, do uso da tecnologia de processos de software.

Contudo, ainda que se reconheça a importância da rastreabilidade como um mecanismo de apoio ao controle de alterações, nem todos os sistemas de GCS provêem funcionalidades para tal, e menos ainda usando mecanismos baseados em gerência de conhecimento. Os sistemas citados acima, *GConf* e Odyssey-SCM, aplicam mecanismos de gerência de conhecimento à GCS.

GConf, via uma interface com *Acknowledge*, a ferramenta de aquisição de conhecimento da Estação TABA, permite o registro e a busca de conhecimento relacionado ao processo de GCS. Integrando *GConf* a *Ackhowledge* é possível registrar idéias, dúvidas, problemas, diretrizes e lições aprendidas. Uma vez registradas, essas informações são filtradas, empacotadas e armazenadas no repositório da organização, vinculadas à atividade do processo de GCS na qual foram geradas, ficando acessíveis aos usuários de *GConf*. Entretanto, ainda que *GConf* faça algum uso de gerência de conhecimento, ela não o faz no sentido de apoiar a avaliação de impacto de alterações.

Já o Odyssey-SCM [Murta 2006] visa fornecer processos e ferramental de apoio para a aplicação de técnicas de GCS no contexto do desenvolvimento baseado em componentes (DBC). A abordagem Odyssey-SCM envolve: (i) adaptações do processo de GCS para o DBC; (ii) controle de modificações com ênfase nos requisitos específicos do DBC; (iii) sistema de controle de versões focado em artefatos de alto nível de abstração e que permite o versionamento de elementos da UML, tais como pacotes, classes, casso de uso etc; (iv) gerenciamento de construção, liberação, empacotamento e implantação de componentes; e integração dos espaços de trabalho de GCS e DBC, que é responsável, inclusive, por propagar conhecimento entre ferramentas de GCS e DBC. Dentre os conhecimentos propagados, estão aqueles relativos a rastros de modificação entre elementos de modelo UML e que são usados, por exemplo, para análise de impacto de alteração. Tal conhecimento é obtido por meio de técnicas de mineração de dados em repositórios do sistema de GCS.

A rastreabilidade em Odyssey-SCM (mais especificamente Odyssey-WI) possui um foco diferente da apresentada neste trabalho, visto que ela se dá apenas entre modelos (e elementos de modelo) UML, enquanto que neste trabalho ela ocorre entre artefatos de qualquer tipo. Apesar disso, ambos possuem similaridades, já que consideram informações armazenadas pelo sistema de GCS para fazer sugestões de dependências. Por outro lado, a detecção de dependências em Odyssey-WI parece ser mais robusta que em GCS-ODE, uma vez que utiliza algoritmos de mineração de dados para tal. Além disso, o mecanismo de rastreabilidade em Odyssey-WI atribui semântica aos rastros proporcionando algumas vantagens. São indicadas juntamente com o rastro, informações coletadas no sistema de GCS que forneçam algum significado semântico a ele, facilitando o entendimento e permitindo, inclusive, avaliar a sua validade. Por sua vez, ODE leva em consideração as informações de alterações em projetos similares para detectar dependências, além de permitir definir um mapa de dependências organizacional e fazer sugestões a partir dele.

6. Conclusões

O controle de alterações é uma parte essencial da Gerência de Configuração de Software (GCS). Quando uma alteração é feita, deve-se garantir que ela não vai comprometer a integridade de outros artefatos. Assim, é muito importante rastrear artefatos dependentes e indicá-los como elementos a serem analisados em uma alteração. Este artigo apresentou uma abordagem baseada em gerência de conhecimento para aperfeiçoar a rastreabilidade de artefatos e apoiar a avaliação de impacto de alterações.

A nova versão da ferramenta de GCS do ambiente ODE, que implementa a abordagem apresentada neste trabalho, está disponível na versão 1.4 do ambiente ODE, a ser disponibilizada para as organizações parceiras em breve (previsão: dezembro/2008). Com a sua efetiva utilização, espera-se poder avaliar os reais benefícios obtidos e dificuldades encontradas, de modo a aperfeiçoar o esquema de sugestões em trabalhos futuros.

Agradecimentos

Este trabalho foi realizado com o apoio da VixTeam Consultoria e Sistemas e da Projeta Sistemas, empresas parceiras que têm financiado o projeto e dado feedback de sua aplicação a casos reais.

Referências

Antoniol, G., Canfora, G., Casazza, G., Lucia, A. (2002) “Recovering Traceability Links between Code and Documentation”. IEEE Transactions on Software Engineering, vol. 28, n. 10.

Arantes, L.O., Falbo, R.A., Guizzardi, G. (2007) “Evolving a Software Configuration Management”, Second Workshop on Ontologies and Metamodeling Software and Data Engineering – WOMSDE´2007, XXI Simpósio Brasileiro de Engenharia de Software – SBES´2007, João Pessoa.

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

Carvalho, V.A., Falbo, R.A., Arantes, L.O. (2006) "EstimaODE: Apoio a Estimativas de Tamanho e Esforço no Ambiente de Desenvolvimento de Software ODE", V Simpósio Brasileiro de Qualidade de Software, SBQS´2006, Vitória – ES, p. 12-26.

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

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.

Falbo, R. A., Arantes, D.O., Natali, A.C.C. (2004) "Integrating Knowledge Management and Groupware in a Software Development Environment". In: 5th International Conference on Practical Aspects of Knowledge Management - PAKM'2004, Vienna, Springer-Verlag Berlin Heidelberg, Vol. 3336, pp. 94-105.

Falbo, R.A., Ruy, F.B., Dal Moro, R. (2005) "Using Ontologies to Add Semantics to Software Engineering Environments". 17th International Conference on Software Engineering and Knowledge Engineering, SEKE'2005, Taipei, China, p. 151-156.

Figueiredo, S., (2004) Gerência de Configuração em Ambientes de Desenvolvimento de Software Orientados a Organização. Projeto de Graduação, UFRJ.

IEEE (2004) *SWEBOK - Guide to the Software Engineering Body of Knowledge*, 2004 Version, IEEE Computer Society.

ISO/IEC (2008), ISO/IEC 12207 – *System and Software Engineering - Software Life Cycle Processes, 2nd edition*.

Lima, K.V.C. (2004) Definição e Construção de Ambientes de Desenvolvimento de Software Orientados a Organização. Tese de Doutorado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ.

Murta, L.G.P. (2006) Gerência de Configuração no Desenvolvimento Orientado a Componente. Tese de Doutorado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ.

Nagel, W. (2005) *Subversion Version Control: Using The Subversion Version Control System in Development Projects*, Prentice Hall PTR.

Nunes, V.B., Falbo, R.A. (2006) "Uma Ferramenta de Gerência de Configuração Integrada a um Ambiente de Desenvolvimento de Software", V Simpósio Brasileiro de Qualidade de Software, Vila Velha - ES.

Pressman, R. S. (2006), *Engenharia de Software*, Mc Graw Hill, 6a edição.

Rus, I., Lindvall, M., (2002) "Knowledge Management in Software Engineering", IEEE Software, vol. 19, no 3, pp. 26-38, May/June.

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

Staab, S., Studer, R., Schurr, H.P., Sure, Y. (2001) "Knowledge Processes and Ontologies", IEEE Intelligent Systems, Vol. 16, No. 1, January/February.

SEI (2006), *CMMI for Development – v1.2*, Pittsburgh: Software Engineering Institute.