Iniciação Científica **CESUMAR**
Jul./Dez. 2011, v. 13, n. 2, p. 159-170

ESTUDO E IMPLEMENTAÇÃO DE UM SISTEMA PARA CONTROLE DE VERSÕES

VILSON RAPHAEL VELLO DE ANDRADE*

ALINE MARIA MALACHINI MIOTTO AMARAL**

RESUMO: A demanda por qualidade tem levado a comunidade de software ao desenvolvimento de inúmeras técnicas e ferramentas que auxiliem o desenvolvimento perante o grande número de itens de configuração envolvidos no decorrer de um desenvolvimento. Para assegurar tal qualidade é necessário o uso de procedimentos e ferramentas que auxiliem o processo de desenvolvimento de software. Dentre tais processos destaca-se ao gerenciamento e controle de versões, cujo estudo foi alvo deste projeto. Baseado nos recursos a serem suportados pela ferramenta S.A.Do.M (*Software Artifacts Document and Management*), em desenvolvimento pelo Grupo de Pesquisa em Sistemas de Informação (GPSI), foi realizado um levantamento das características que devem ser implementadas em um gerenciador de versões a ser integrado a esta ferramenta. Deve-se destacar que a S.A.Do.M tem como objetivo principal o armazenamento e o gerenciamento da documentação de artefatos de software.

PALAVRAS-CHAVE: Artefatos de Software; Controle de Versões; Gerenciamento de Configuração de Software.

IMPLEMENTATION OF A SYSTEM FOR THE CONTROL OF SOFTWARE VERSIONS

ABSTRACT: Quality demands have led the software community to develop techniques and tools that assist the development of several configuration items involved in the development process. Procedures and tools that aid the process of software development are required to warrant such quality. The management and version control may be highlighted and exposed in current analysis. Based on features supported by the tool Software Artifacts and Document Management (S.A.Do.M), developed by the Research Group on

* Discente do Curso de Sistemas de Informação no Centro Universitário de Maringá – CESUMAR; Bolsista do Programa de Bolsas de Iniciação Científica do Centro Universitário de Maringá – PIBIC/CNPq - Cesumar). E-mail: raphael@raphaelvello.com.br

** Docente dos cursos de Análise e Desenvolvimento de Sistemas, Sistemas de Informação e Sistemas para Internet do Centro Universitário de Maringá – CESUMAR. E-mail: amiotto@cesumar.br

Information Systems (GPSI), a survey was undertaken on the characteristics that should be implemented in a release manager to be integrated into this tool. Main aim of S.A.Do.M is the storage and management of software artifacts documentation.

KEYWORDS: Software Artifacts; Version Control; Software Configuration Management.

INTRODUÇÃO

Os níveis de complexidade e abstração exigidos pelos atuais sistemas impactam diretamente nos esforços empenhados para realizar a sua documentação (BRUEGGE et al., 2006). Considerando as constantes mudanças que afetam as regras do mercado e também as necessidades dos usuários, torna-se difícil predizer como um software evoluirá com o passar do tempo (PRESSMAN, 2001). Nesse contexto de modificação constante e alta complexidade dos artefatos de software produzidos, a Gerência de Configuração de Software (GCS) é a área da Engenharia de Software cujo principal objetivo é evitar a perda de controle do projeto do ponto de vista da gerência de artefatos. A GCS também pode ser definida como o aspecto do processo de gerência que foca exclusivamente em controlar as mudanças que ocorrem durante o processo (ISO/IEC12207, 1995). Um dos procedimentos que se destaca durante o desenvolvimento de software é o controle de versões, sendo este uma das principais funcionalidades do Gerenciamento de Configuração de Software. A falta de um correto controle de versões dos artefatos de software gerados impacta diretamente na correta elaboração da documentação do software produzido.

O controle de versões combina procedimentos e ferramentas para gerenciar diferentes versões que são criadas durante o processo de engenharia de software, preocupando-se com a manutenção da rastreabilidade das versões de um sistema, coordenando o trabalho paralelo de desenvolvedores por meio das mais diferentes funcionalidades.

De acordo com o relatório sobre Qualidade e Produtividade no Setor de Software Brasileiro (BRASIL, 2002) referente a 2001, somente 10,4% das empresas entrevistadas (45 empresas de um espaço amostral de 431) praticavam gestão de mudanças. Além disso, somente 20,1% das empresas entrevistadas (85 empresas de um total de 423) faziam uso de ferramentas de apoio ao GCS.

Segundo Oliveira (2005), flexibilidade constitui-se no reconhecimento de que dois projetos de desenvolvimento de software não são idênticos. Desta forma, devem ser fornecidos mecanismos para que os gerentes de configuração possam moldar, tanto quanto possível, a abordagem de controle de versões conforme suas necessidades. a abordagem de controle de versões conforme suas necessidades.

Em função da necessidade de ferramentas que gerenciem a produção de software de forma flexível, adequando-se à necessidade de cada desenvolvimento e controlando seus artefatos de software, o Grupo e Pesquisas em Sistemas de Informação[1] (GPSI) desenvolve, por meio de pesquisas cientificas, estudos que visam à implementação de uma ferramenta para documentação e gerenciamento dos artefatos produzidos ao longo do processo de software, chamada de S.A.Do.M (*Software Artifacts Documentation Manager*). Um dos fatores relevantes para o bom funcionamento de tal ferramenta relaciona-se diretamente a um eficaz controle e gerenciamento de versões que deve envolver a sincronização de mudanças, armazenamento

[1] Grupo de Pesquisas em Sistemas de Informação (GPSI) – Grupo destinado a pesquisas na área de sistemas de informação do Centro Universitario de Maringá - CESUMAR

de histórico de mudanças, comunicação com o repositório e recuperação de revisões para os mais variados tipos de artefatos.

Dentro deste contexto, este trabalho teve como objetivo realizar um estudo detalhado das características necessárias para um suporte efetivo ao gerenciamento das versões dos artefatos que serão armazenados pela S.A.Do.M.

Em uma primeira etapa foi realizado o levantamento e estudo de todas as características e terminologias envolvidas durante o gerenciamento de versões e revisionamento de artefatos de software. Formou-se uma sólida base de conhecimento sobre o tema abordado nesta pesquisa possibilitando em uma segunda etapa realizar o levantamento bibliográfico no que diz respeito ao conceito de gerenciamento de configuração de software e controle de versões. Buscou-se as melhores práticas e modelos com uma análise primária das ferramentas hoje existentes assim como a evolução de diversas outras. Após tais estudos, foram levantadas as necessidades da ferramenta S.A.Do.M assim como seu modelo, que foi objeto de outro projeto de pesquisa executado pelo GPSI. Com base neste levantamento foi elaborado o modelo para implementação do gerenciador de versões suportado pela ferramenta S.A.Do.M

O presente artigo está organizado da seguinte forma: na seção 2 é apresentada uma visão geral sobre o controle de versões de artefatos, cujos princípios nortearem esta pesquisa; na seção 3 são descritos os principais fundamentos relacionados ao gerenciamento de versões e na seção 4 são abordados o modelo de dados da S.A.Do.M e as características de gerenciamento de versões que deverão ser suportadas nesta ferramenta. Finalmente na seção 4 são apresentadas as considerações finais deste trabalho.

2 CONTROLE DE VERSÕES

2.1 CONCEITOS SOBRE CONTROLE DE VERSÕES

Diferentes abordagens para o controle de versões foram propostas nos últimos anos, mas a principal diferença entre elas é a sofisticação de atributos que são usados para construir versões específicas e variações de um sistema e o mecanismo de processo para a construção (PRESSMAN, 2001).

Controle de versões combina procedimentos e ferramentas para gerenciar diferentes versões de objetos de configuração que são criados durante o processo de software, as quais são identificadas por meio de atributos específicos. Um exemplo de atributo é o número da versão que é guardado juntamente com o objeto (OLIVEIRA, 2007).

O termo versionamento implica em suporte ao desenvolvimento paralelo, por meio do qual o sistema evolui em diferentes variações para a adequação do software a diferentes ambientes de operação, satisfazendo assim os requisitos do usuário (CHAN; HUNG, 1997).

O controle de versões em uma ferramenta deve prover mecanismos de ramificação (*branching*), fusão (*merge*) e trancamento (*locking*), os quais são inerentes ao conceito de versionamento, proporcionando comparação e impressão de diferenças em versões do mesmo item de configuração (CHAN; HUNG, 1997). Para um eficiente controle de versões a ferramenta deve suportar os seguintes mecanismos:

- Ramificação (*branching*): Mudanças em um objeto de configuração podem ocorrer paralelamente a outras modificações e não seqüencialmente. Neste sentido o mecanismo de ramificação provê a divisão de um determinado item ou conjunto de itens em versões paralelas (SCHAMP, 1995).
- Fusão (*merge*): Da mesma forma que um item ou conjunto de itens são divididos, eles também precisam ser unidos através do mecanismo de fusão (*merge*), conciliando as alterações realizadas nos caminhos paralelos (SCHAMP, 1995).
- Diferenciação (Diff): Compara diferentes versões de um mesmo objeto, muito útil para

analisar mudanças realizadas em sistemas complexos (SCHAMP, 1995).

- Compressão: As ferramentas de controle de versões devem prover esse mecanismo para o melhor uso do espaço utilizado para o armazenamento de versões de itens.
- *Check in/ check out*: Antes de começar a trabalhar em um item, o desenvolvedor executa um *check out*. Esta operação copia a versão selecionada do repositório para seu local de trabalho. A partir daí o desenvolvedor pode realizar alterações na versão sem interferir nas atividades de outros desenvolvedores. O mecanismo de *check out* assegura alguns direitos para o desenvolvedor para evitar alterações inadvertidamente sobrescritas nas mudanças de outros. O *check in* salva o item no repositório novamente (ESTUBLIER et al., 2002). Este mecanismo em ferramentas provê bom gerenciamento de itens de configuração individuais, provendo o controle do repositório e suporte à coordenação de espaços de trabalho de usuários (CHAN; HUNG, 1997).
- Hierarquia: Provê a organização dos arquivos em grupos que suportam as visões dos projetos (CHAN; HUNG, 1997);
- Controle de liberações: Identifica versões macro do sistema. Esta característica uma ferramenta é importante para esclarecer sobre o impacto das mudanças entre as versões (SCHAMP, 1995).
- Marcação de versões: Realiza as marcações de liberações do software, sendo de grande importância para paralelismo em um ambiente (SCHAMP, 1995).
- Armazenamento de diferentes tipos arquivos: A ferramenta deve permitir o versionamento de arquivos com diferentes tipos de dados e não apenas textuais (SCHAMP, 1995).
- Controle de acesso: Limita ações de usuários não autorizados. Um modelo de controle de acesso mais robusto provê a definição de privilégios diferenciados entre usuários (SCHAMP, 1995). Os níveis de autoridade no acesso variam dependendo do tipo do item de configuração e do impacto que a mudança produz no sistema. Por exemplo, mudanças no código durante o ciclo de desenvolvimento de um software normalmente requerem um baixo nível de autorização, mas mudanças no mesmo código depois de ele ter sido liberado para uso geral requer um nível mais alto de autorização. Outro exemplo é que mudanças ocorridas em versões rascunho de documentos são menos controladas que mudanças nas versões finais (IEEE, 1987).
- Relacionamentos entre itens: Possuem atributos e relacionamentos com outros objetos do repositório. Os relacionamentos permitem verificar o que pode ser afetado se uma mudança no item ocorrer.

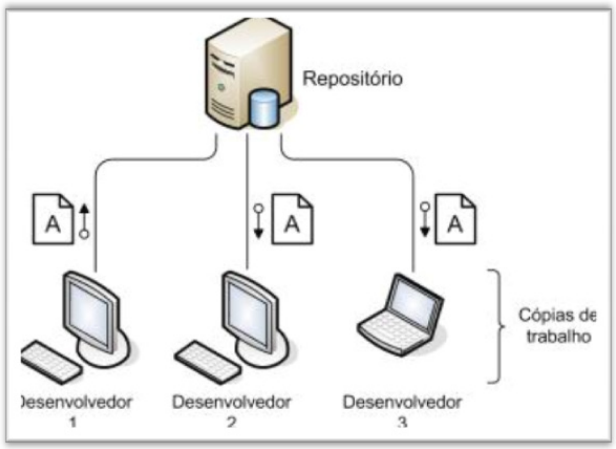

Figura 1 Funcionamento básico do processo de Controle de Versões

Com o andamento e crescimento do projeto, o repositório passa a guardar múltiplas versões dos artefatos que são pertinentes a um projeto. Essas múltiplas versões são organizadas em revisões.

Uma revisão é definida com todos os artefatos que compõem um projeto em um determinado momento. Esta situação pode ser observada na figura 2.

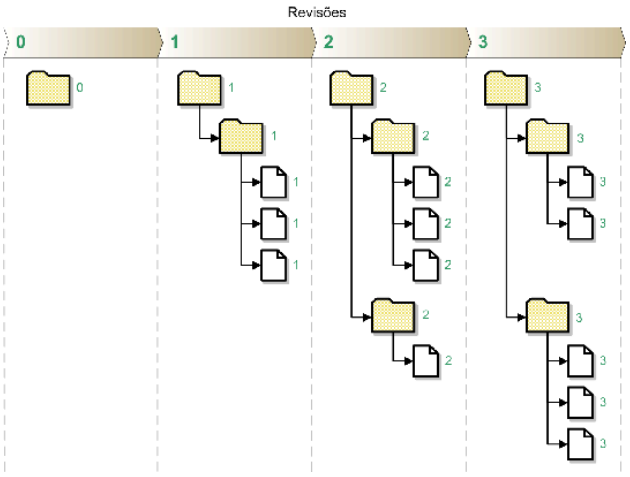

Figura 2 Exemplo de uma arvore hierárquica de diferentes versões de arquivos

Para economizar espaço, apenas as diferenças entre as revisões costumam ser armazenadas no repositório. Quando se deseja recuperar determinado artefato, as diferenças são analisadas e o arquivo é remontado de acordo com a revisão desejada.

Como várias pessoas trabalham em paralelo e concorrentemente nos mesmos artefatos do projeto, é necessária uma política para ordenar e integrar todas essas alterações, de modo a evitar que uma pessoa sobrescreva o trabalho de outra.

Além de rastrear e controlar alterações, um sistema de controle de versões também coordena a edição colaborativa e o compartilhamento de dados entre várias pessoas envolvidas em um projeto.

2.1.1 Cópia de Trabalho

Os desenvolvedores não trabalham diretamente nos artefatos do repositório. Cada desenvolvedor possui uma cópia de trabalho que espelha os artefatos do repositório para trabalhar com liberdade enquanto termina suas tarefas, isolado dos outros desenvolvedores.

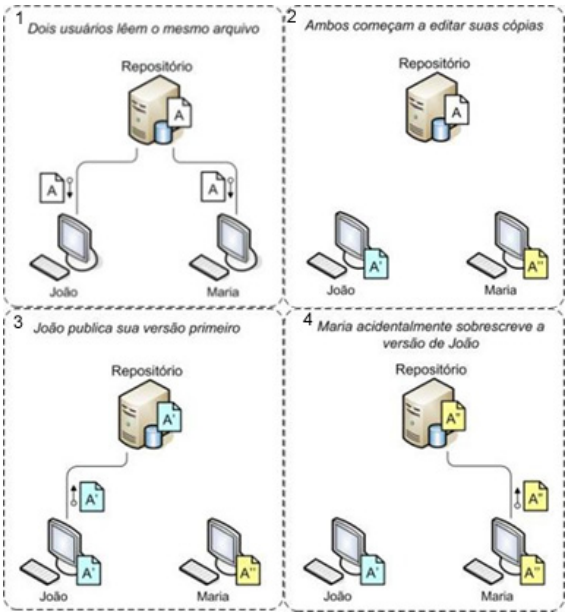

Figura 3 Exemplificação do problema de compartilhamento.

Ao final de ambas as alterações, mesmo que a versão do usuário "João" não seja perdida (já que o sistema de controle de versões se lembra de cada alteração), na prática as mudanças deste usuário não serão consideradas e estarão acidentalmente ausentes na última versão do arquivo.

2.1.2 Trava-Modifica-Destrava

O sistema de controle de versões permite que apenas uma pessoa por vez altere um determinado arquivo, como pode ser observado na figura abaixo.

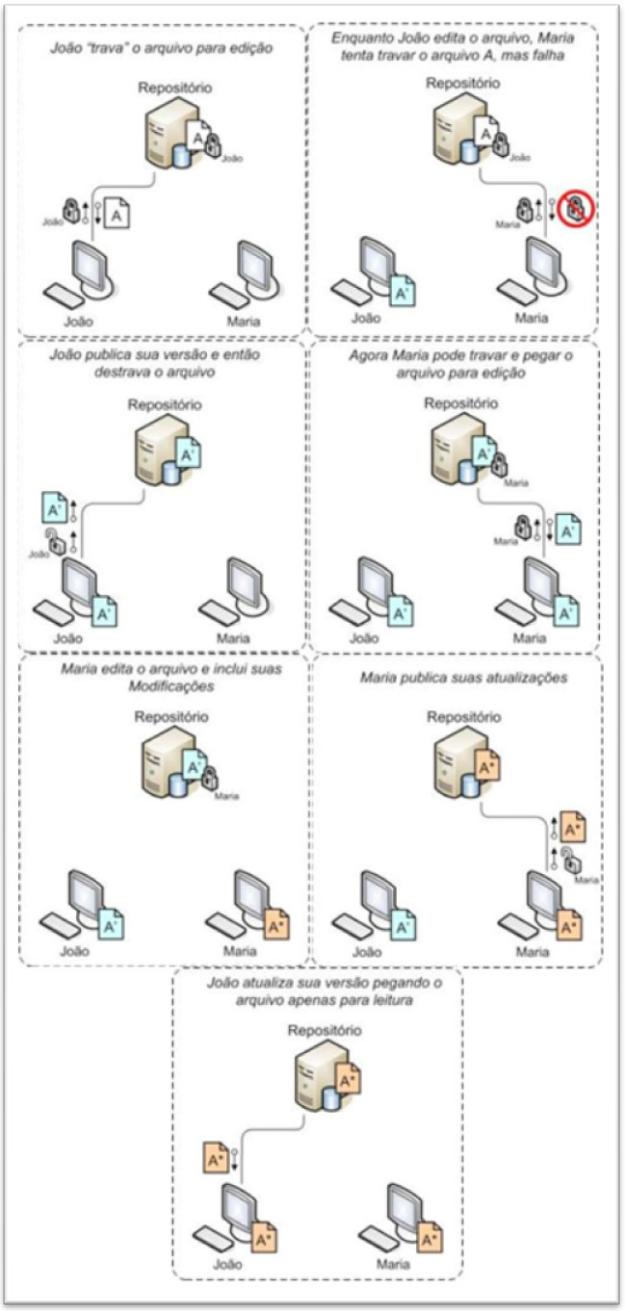

Figura 4 Exemplificação da solução trava-modifica-destrava.

2.1.3 Copia-Modifica-Resolve

Neste modelo não existe travamento de artefatos. Cada desenvolvedor trabalha livremente em qualquer arquivo da sua cópia de trabalho. Ao final, as alterações de cada desenvolvedor são mescladas no repositório formando a versão final, conforme pode ser observado na figura 5.

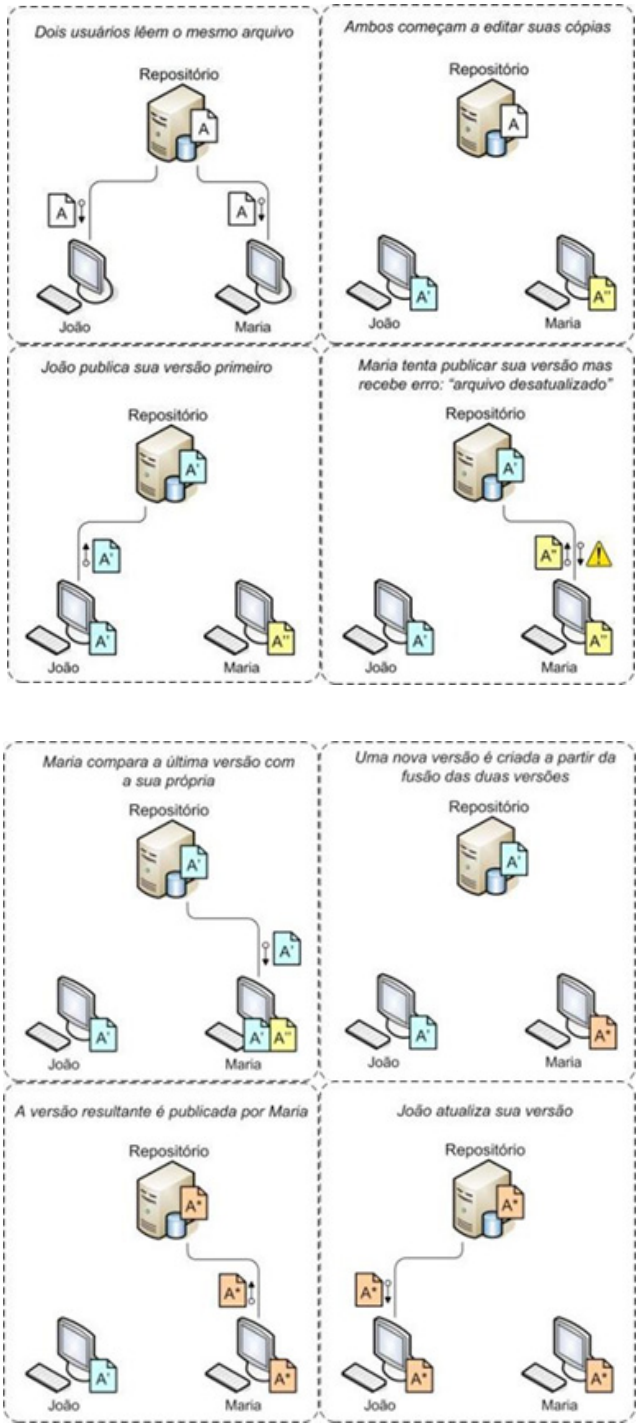

Figura 5 Exemplificação da solução copia-modifica-resolve.

2.1.4 Considerações finais sobre controle de versões

A solução "trava-modifica-destrava" é restritiva e frequentemente atrapalha o trabalho dos usuários. O travamento pode causar alguns problemas

administrativos e forçar a uma serialização desnecessária.

A solução "copia-modifica-resolve" parece um pouco caótica a princípio, mas funciona bem na prática. Conflitos (sobreposições de alterações nos mesmos trechos de um arquivo) são raros e são causados basicamente pela falta de comunicação entre desenvolvedores. Na grande maioria dos casos as alterações não se sobrepõem e são mescladas automaticamente pelo sistema de controle de versão.

Como o repositório registra todas as alterações efetuadas, o sistema de controle de versão pode informar com facilidade quais as alterações que foram realizadas entre uma revisão e outra, quando isso ocorreu e quem fez as alterações. Essas informações permitem um controle maior sobre mudanças no projeto.

O controle de versões resolve diversos problemas básicos de desenvolvimento tais como uso de diferentes versões de código, sincronização do trabalho paralelo de desenvolvedores no mesmo projeto, recuperação de versões anteriores e registro do histórico de alterações.

Um controle de versões de software é uma prática de engenharia de software comprovadamente eficaz. Por isso, faz parte das exigências para melhorias do processo de desenvolvimento de certificações tais como CMMI e SPICE.

3 GERENCIAMENTO DE VERSÕES

3.1 BREVE HISTÓRICO

Diversos autores propuseram as mais variadas abordagens para a pratica de gerenciamento de versões. Tais abordagens preocupam-se em gerenciar tipos específicos de objetos, ou modelos de dados altamente sofisticado (OLIVEIRA, 2007).

Abordagens como a de (HABERMANN; NOTKIN, 1986), através da árvore sintática da linguagem C, propôs (específico ao desenvolvimento baseado em C) um versionamento orientado a estrutura, como vantagem o sistema conseguia manter uma consistência na sintaxe do código fonte gerenciado. Já (GOLDSTEIN; BOBROW, 2009) exibiram em 1980 uma abordagem voltada a linguagem Smaltalk; e (RENDER; CAMPBELL, 1991) tiveram seu modelo de dados baseado em uma proposta de controle de versões focada em orientação a objetos, com seu sistema escrito em Smaltalk, versionavam códigos escritos em Pascal.

Pode-se notar nos exemplos apresentados acima a limitação de uso de tais abordagens, levando em conta a necessidade de uso de linguagens de programação ou metodologias de trabalho específicas.

3.2 FERRAMENTAS DE GERENCIAMENTO DE VERSÕES

Atualmente existem diversas ferramentas que apóiam o desenvolvedor de software no processo de gerenciamento de versões de seus artefatos produzidos, trazendo agilidade ao desenvolvimento e conseqüentemente possibilitando uma melhoria do produto final. No entanto, nem todas atendem de forma flexível projetos que possuam variados tipos de artefatos e metodologias especificas. De todas as ferramentas encontradas e estudadas, todas focam em um determinado tipo de desenvolvimento ou artefato.

Dentre as ferramentas mencionadas, pode-se citar a *Borland TeamSource*, *Intersolv PVCS Version Manager*, *Microsoft Visual SourceSafe*, *Rational Clear Case* e *Concurrent Version System* entre muitas outras. Para esta pesquisa foram levantadas as características de uma ferramenta predecessora e tomada como referência por muitas outras, o CVS (CAETANO, 2004), o qual possui seu código fonte aberto e pouca restrição quanto ao seu modo de utilização. Porém, para foco deste estudo adotou-se o SubVersion

(COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2007), resultado de inúmeras melhorias e considerado sucessor do CVS.

Dentro do contexto da ferramenta S.A.Do.M, foram levantadas as características de um gerenciador de versões para a mesma, que possibilite ao desenvolvedor e sua equipe gerenciar os mais diversos tipos de artefatos de software, independente do seu conteúdo, preocupando-se apenas com sua metodologia e permitindo flexibilidade à equipe de desenvolvimento, permitindo adaptar a metodologia a uma existente ou criando metodologias conforme a necessidade do projeto.

3.3 FERRAMENTA SUBVERSION

Diversas características determinaram a escolha de uma ferramenta já existente para servir de base aos estudos e futura implementação do mecanismo de gerenciamento de versões da S.A.Do.M

Dentre as características do SubVersion que mais contribuíram para a sua escolha, pode-se citar a forma com que esta registra o histórico de mudanças de árvores de diretório inteiras ao longo do tempo, ou seja, o SubVersion controla não só as versões dos arquivos, como também dos diretórios. Outra função relevante do SubVersion é a possibilidade de adicionar, excluir, copiar e renomear tanto arquivos quanto diretórios. Cada novo item adicionado tem um histórico próprio, limpo e novo. Ao contrário do que acontece no CVS, o SubVersion suporta que quando enviadas conjuntamente modificações (que pode envolver diversos arquivos e diretórios), as mesmas sejam registradas no repositório de forma atômica.

O SubVersion possui um mecanismo por meio do qual é possível atribuir informações diversas (nomeadas de *properties)* aos artefatos, sem que seja necessário armazená-las no interior do repositório. Cada artefato pode possuir uma ou mais propriedades. Mesmo havendo propriedades pré-definidas pelo SubVersion, é possível criar e armazenar propriedades personalizadas, não havendo um limite para a quantidade de propriedades por artefato.

3.4 MODELO DA FERRAMENTA S.A.DO.M PROPOSTO PELO GPSI

Durante diversas reuniões realizadas pelo GPSI, debateu-se acerca dos diversos recursos que a ferramenta S.A.Do.M deveria suportar, chegando à concepção do modelo de dados representado pela figura 6.

Após a elaboração do modelo de dados, definiu-se que as entidades Projeto, Artefato e Artefato Vinculado seriam alvos diretos do sistema de gerenciamento de versões e, a fim de apoiar outras necessidades da ferramenta S.A.Do.M, identificaram-se alguns requisitos primários, tais como:

- Estrutura do armazenamento dos artefatos a serem documentados à esta será feita em repositório centralizado.
- Método de controle e acesso a versões de artefatos à deverá ser do tipo *Lock-Modify-Unlock* evitando conflitos em alterações por mais de um desenvolvedor.
- Layout de repositórios à deverá ser do tipo compartilhado, permitindo que determinado artefato possa compor mais de um projeto.

Com esta base formada e tendo em vista os propósitos e necessidades expostos nos estudos realizados pelo GPSI no decorrer da concepção da ferramenta S.A.Do.M apresenta-se como trabalho futuro o desenvolvimento, de forma integrada à ferramenta S.A.Do.M, da ferramenta de controle de versões que apóie a documentação de software.

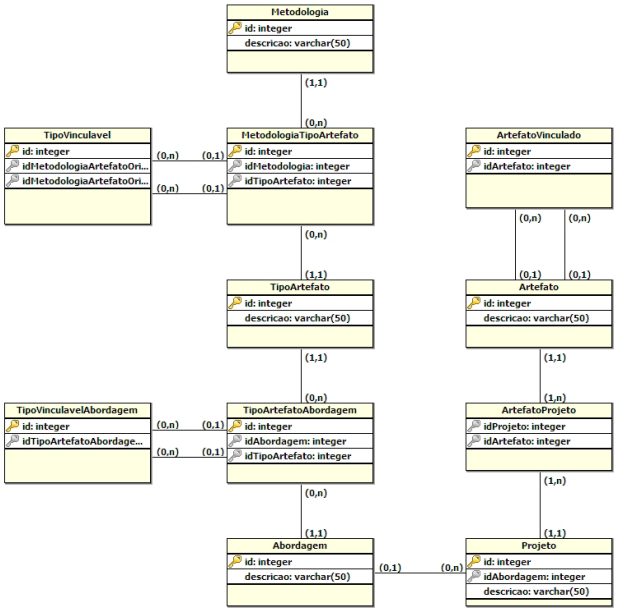

Figura 6 Modelo lógico de dados da ferramenta S.A.Do.M

3.5 CARACTERÍSTICAS DE GERENCIAMENTOS DE VERSÕES A SEREM SUPORTADOS NA S.A.DO.M

Apoiado pelos estudos realizados e pelo modelo de dados apresentado na figura 6 são apresentadas a seguir as principais características do gerenciador de versões da ferramenta S.A.Do.M.

3.3.1 Relacionamento das versões dos artefatos e estado dos objetos

As versões de um mesmo artefato estão ligadas por meio de um relacionamento de derivação linear. Cada versão é criada como sucessora de outra, e esta é considerada sua antecessora (NORONHA, 2000). Informações relevantes para a localização de um determinado objeto deverão ser suportadas pela ferramenta S.A.Do.M por meio de metadados associados.

Ao término do procedimento de *check-in* de qualquer artefato, o repositório armazena informações sobre as alterações em um banco de dados. Tais informações propiciam um histórico relevante para toda a equipe de desenvolvimento e apóiam a resolução de qualquer problema que possa surgir decorrente de tais modificações. Além das informações relativas às alterações efetuadas, devem-se armazenadas as informações sobre data e hora de tais modificações assim como o responsável por estas.

3.3.2 Estado dos artefatos

Para cada versão também é estabelecido um atributo de status identificador de quais operações poderão ser realizadas sobre este artefato. Este status poderá ter os seguintes valores: *estável*, *em trabalho* ou *out of version*. Uma versão possui status "em trabalho" enquanto estiver sendo modificada. Quando esta versão estiver pronta, ou seja, enquanto quem efetuou o *check-out* deste artefato não efetuar um *check-in*. Após tal processo de *check-in*, a versão é definida como estável e a versão anterior passa ter seu status "out of version".

A versão "estável" é a versão corrente (mais atual) do artefato, não podendo ser alterada, até que ela mude novamente de status. Podem-se remover versões que possuam status "em trabalho" ou "estáveis", desde que essas não possuam sucessoras, porém estas não terão versões anteriores e rastreáveis.

3.3.3 Armazenamento do histórico de alterações

Quando um desenvolvedor efetua alterações em um artefato, o repositório armazena em um banco de dados as informações sobre estas atualizações. Estes dados propiciam um importante histórico para a solução de problemas decorrentes da manutenção de artefatos. Através deste histórico,

obtêm-se informações pertinentes às alterações efetuadas agilizando o acesso a dados, tais como: qual desenvolvedor que efetuou a manutenção, data/hora em que ocorreram e a descrição informada.

3.3.4 Restrições de acesso

O desenvolvedor associado a um projeto específico pode executar as tarefas de *get*, *lock*, *unlock*, *check-out*, *check-in*, adição de artefato e voltar versão. Já o administrador do repositório pode efetuar todas as operações do desenvolvedor, além de remover artefatos, voltar qualquer versão e criar projetos deliberando responsabilidades aos desenvolvedores.

4 CONSIDERAÇÕES FINAIS

Atualmente, dentro do contexto de produção de software, as atividades de GCS e, mais especificamente, as atividades de controle de versões orientam os desenvolvedores durante a evolução do software e permitem um controle e recuperação de inúmeras informações sobre os artefatos controlados, sendo parte fundamental no processo de desenvolvimento.

A qualidade de software, tão buscada nos atuais dias, está intrinsecamente ligada à qualidade dos processos utilizados para o desenvolvimento. Com base neste fato, a melhoria da qualidade de software é resultado direto da melhoria da qualidade dos processos.

A documentação de um software é extremante fundamental para assegurarmos sua qualidade. Para termos uma documentação completa e bem definida temos que garantir que durante o processo de software teremos ferramentas que nos auxiliarão na tarefa de gerenciar, documentar e armazenar tal documentação bem como também controlar todas as mudanças ocorridas em tal documentação. Diante de tais fatores e baseado nos estudos realizados, foram definidas as principais características que deverão ser suportadas no gerenciamento de versões da ferramenta S.A.Do.M, tais como: suporte a um repositório distribuído, permitir diferentes políticas de trabalho conforme o tipo do projeto e conforme o modelo de versionameto e suporte à comunicação por rede e web de forma ágil.

REFÊRENCIAS

BRUEGGE, B. et al.. Supporting Distributed Software Development with fine-grained Artefact Management. In: INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING – IEEE. 2006. Anais... Washington, DC: IEEE Computer Society, 2006.

CAETANO, C.. **CVS Controle de versões e desenvolvimento colaborativo de software**. [S. l.]: Novatec, 2004.

CHAN, A. K. F.; HUNG, S.. **Software Configuration Management**. [S. l.]: [S. n.], 1997.

COLLINS-SUSSMAN, B.; FITZPATRICK, B. W.; PILATO, C. M.. **Controle de Versão com Subversion**. [S. l.]: [S. n.], 2007.

ESTUBLIER, J. ET AL.. IMPACT OF THE RESEARCH COMMUNITY ON THE FIELD OF SOFTWARE CONFIGURATION MANAGEMENT: summary of an impact project report. **ACM SIGSOFT SOFTWARE ENGINEERING NOTES,** V. 27, N. 5, P. 31-39, SET. 2002.

GOLDSTEIN, I. P.; BOBROW, D. G.. **A Layered Approach to Software Design.** Palo Alto. Disponivel em:<http://www.scribd.com/doc/4814413/A-Layered-Approach-to-Software-Design-Ira-P-Goldstein-and-Daniel-G-Bobrow>. Acesso em: maio 2009.

HABERMANN, N.; NOTKIN, D.. **Gandalf:**

Software Development Environments. In: IEEE Transactions on Software Engineering (TSE). [S. l.]: [S. n.], 1986.

IEEE. IEEE Guide to Software Configuration Management. [S. l.]: [S. n.], 1987.
ISO/IEC12207. **Information Technology**. [S. l.]: [S. n.], 1995.

BRASIL. MINISTÉRIO DA CIÊNCIA E TECNOLOGIA - MCT. **Qualidade e Produtividade no Setor de Software Brasileiro**. Brasilia, DF: MCT/SEPIN, 2002.

NORONHA, M. A.. **Um mecanismo para controle de versões num sistema de documentos.** Porto Alegre, RS: [S. n.], 2000.

OLIVEIRA, H. L. R. D.. **ODYSSEY - VCS:** Uma abordagem para controle de versões para elementos da UML. Rio de Janeiro, RJ: [S. n.], 2005.

OLIVEIRA, V. N. P. D.. **Requisitos de Ferramentas de Gerenciamento de Configuração**. Belo Horizonte, MG: [S. n.], 2007.

PRESSMAN, R. S.. **Software Engineering -** A practitioner's Approach. [S. l.]: McGraw-Hill, 2001.

RENDER, H.; CAMPBELL, R.. **An object-oriented model of software configuration management.** [S. l.]: IEEE: Institute of Electrical and Electronics Engineers, 1991.

SCHAMP, A.. **CM-Tool Evaluation and Selection**. [S. l.]: [S. n.], 1995.

Recebido em: 17 Setembro 2009
Aceito em: 16 Agosto 2011