Um Processo de Implantação de Gerência de Configuração na Indústria

Cláudia Werner[1], Leonardo Murta[2], Chessman Corrêa[1], Rodrigo Santos[1], João Gustavo Prudêncio[1], Paula Fernandes[1], Marcelo Schots[1], Rafael Cepêda[1], Flávio Lyra[3]

[1] PESC/COPPE, Universidade Federal do Rio de Janeiro (UFRJ)

[2] Instituto de Computação, Universidade Federal Fluminense (UFF)

[3] Centro de Pesquisas de Energia Elétrica (CEPEL)

{werner,chessman,rps,gustavo,paulacibele,schots,rcepeda}@cos.ufrj.br, leomurta@ic.uff.br, flavio@cepel.br

Abstract. *Maintenance represents an important activity in software industry because it is the one that takes the biggest effort among Software Engineering activities, besides its high cost. In this sense, software configuration management helps to break some difficulties related to software maintenance, such as the lack of product knowledge and negligence in maintenance activities. This paper presents a configuration management deployment process based on the standardization of the use of practices and tools, taking into account the software development organizational culture. The results of its application in an industrial case are discussed.*

Resumo. *A atividade de manutenção ganha destaque na indústria de software por exigir o maior esforço entre as atividades de Engenharia de Software, além dos altos custos financeiros. Nesse sentido, a gerência de configuração contribui para romper algumas das dificuldades de manter produtos de software, tais como falta de conhecimento sobre o produto e negligência na condução de mudanças no software. Este artigo apresenta um processo de implantação de gerência de configuração, baseado na uniformização do uso de práticas e ferramentas, e na preocupação com a cultura organizacional de desenvolvimento de software. Os resultados de sua aplicação na indústria são discutidos.*

1. Introdução

A manutenção possui um papel importante no ciclo de vida do software. No início dos anos 90, muitas organizações alocavam no mínimo 50% de seus recursos financeiros para a atividade de manutenção [SCHACH & TOMER, 2000] e, na década atual, esse valor tem chegado a 70%, tornando-se um dos desafios da Engenharia de Software (ES) [BHATT *et al.*, 2004]. De acordo com POLO *et al.* (1999), a razão desse custo elevado se deve, em parte, à natureza dessa atividade, caracterizada pela imprevisibilidade. Isso evidencia a importância das pesquisas sobre o processo de manutenção e sua aplicação na indústria. No entanto, PADUELLI & SANCHES (2006) afirmam que há uma carência de trabalhos que visam documentar as áreas de problemas na atividade de manutenção de software.

KAJKO-MATTSSON *et al.* (2001) justificam as dificuldades dessa atividade por meio dos seguintes pontos: (1) problemas em diagnóstico de erros, relativos a processos investigativos para detectar a real causa do problema antes de corrigi-lo; (2) falta de documentação; (3) falta de conhecimento sobre o produto; (4) negligência na condução de mudanças no software, isto é, falta de gerência; e (5) baixa estima dos mantenedores, decorrente dos pontos anteriores. Visando contribuir para sanar (ou amenizar) estes pontos, a gerência de configuração (GC) no contexto da ES consiste na disciplina que permite evoluir os diferentes artefatos produzidos ao longo do processo de construção do software de forma controlada e organizada [ESTUBLIER, 2000]. A utilização de seus princípios e práticas contribui para que metas de qualidade sejam alcançadas e restrições de tempo sejam respeitadas. Contudo, apesar da importância da GC como apoio à manutenção, menos de 25% das empresas brasileiras de um universo de 2.500 empresas entrevistadas a utilizavam em 2005 [MCT, 2006].

Conforme MURTA (2006), é importante notar que a GC é parte integrante das principais iniciativas de melhoria de qualidade do processo e do produto, tais como ISO 9000, CMMI, ISO/IEC 15504 e MPS.BR. Assim, diversas organizações que produzem software para auxiliar suas atividades, sejam estas atividades meio ou até mesmo atividades fim do negócio, vêm constantemente implantando as práticas de GC dentro sua cadeia produtiva. Nesse contexto, este artigo descreve um processo para implantação de GC baseado na uniformização do uso de práticas e ferramentas, e na preocupação com a cultura organizacional de desenvolvimento de software por meio de uma breve análise da adequabilidade da solução tecnológica a ser adotada em reuniões com os *stakeholders* do projeto. Além disso, é discutida a execução desse processo no Centro de Pesquisas de Energia Elétrica (CEPEL) ao longo dos últimos três anos.

Além desta seção introdutória, este artigo está organizado em outras quatro seções: a Seção 2 apresenta uma visão geral sobre GC, bem como suas funções e os sistemas que a compõem; a Seção 3 descreve o processo proposto para a implantação de GC; a Seção 4 discute o estudo de caso realizado no CEPEL, abrangendo uma breve caracterização dos projetos e algumas das dificuldades encontradas; por fim, a Seção 5 contém as lições aprendidas e algumas considerações finais.

2. Gerência de Configuração

A GC surgiu nos anos 50 para atender a uma necessidade da indústria aeroespacial norte-americana: controlar as modificações na documentação referente à produção de aviões de guerra e naves espaciais [LEON, 2000]. Nas duas décadas seguintes, o seu escopo foi ampliado, passando a abranger também artefatos de software [CHRISTENSEN & THAYER, 2002]. No entanto, o foco ainda permaneceu em aplicações militares e aeroespaciais. Somente com o surgimento da norma na primeira versão da IEEE Std 828 [IEEE, 2005], no início dos anos 80, a GC teve seu foco ampliado a outros domínios.

O uso de sistemas de GC durante o desenvolvimento de software permite obter um nível adequado de controle sobre os artefatos gerados e modificados pelos diferentes *stakeholders* desse processo. A GC não se propõe a definir quando e como devem ser executadas as modificações nos produtos de trabalho, papel este reservado ao processo

de desenvolvimento. A sua atuação ocorre como processo auxiliar de controle e acompanhamento [SOFTEX, 2009].

Sob uma perspectiva gerencial, GC é dividida em cinco funções, que são [IEEE, 2005]: identificação da configuração, controle da configuração, contabilização da situação da configuração, avaliação e revisão da configuração, e gerenciamento de liberação e entrega. Por outro lado, sob uma perspectiva de desenvolvimento, a GC é dividida em três sistemas principais, a saber: (1) controle de modificações: armazena informações sobre solicitações de modificação, relatando-as aos interessados e autorizados; (2) controle de versões: identifica itens de configuração de forma que os mesmos evoluam de forma distribuída e concorrente, porém disciplinada; e (3) gerenciamento de construção: automatiza o processo de transformação dos produtos de trabalho no sistema executável propriamente dito, estruturando também as *baselines* selecionadas para liberação. É importante ressaltar que estas perspectivas não se relacionam de forma complementar, mas sobreposta, ou seja, as cinco funções descritas na perspectiva gerencial podem ser implementadas pelos três sistemas descritos na perspectiva de desenvolvimento, acrescidos de procedimentos manuais, se necessário.

3. Processo Proposto para Implantação de GC

Em organizações que possuem o desenvolvimento de software como uma de suas atividades-fim, a utilização de planos e processos bem definidos auxilia na padronização das atividades e dos artefatos por elas gerados. Por meio da sistematização dos passos necessários à execução das atividades, aumenta-se o controle sobre o trabalho e sobre os dados produzidos, de maneira que isto ajuda que os resultados obtidos sejam previsíveis e repetíveis. Neste sentido, é proposto um processo para a implantação de GC conforme ilustrado na Figura 1.

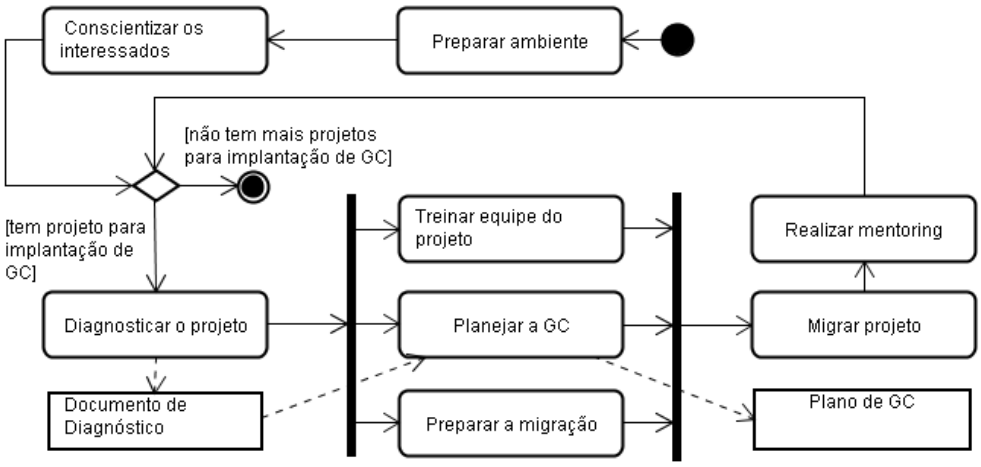

Figura 1 – Processo posposto para implantação da GC

A primeira atividade do processo é **preparar o ambiente**. Esta atividade possui duas tarefas: escolha das tecnologias para GC e implantação das mesmas na organização. A escolha pode ser determinada pelas características dos projetos, tamanho das equipes e recursos oferecidos pelos sistemas de GC.

A segunda atividade do processo é **conscientizar os interessados**. Nesta atividade, os *stakeholders* são devidamente informados sobre as mudanças que serão realizadas. Os motivos e os benefícios a serem obtidos são explicitados. Outros objetivos dessa atividade são manter todos os envolvidos motivados e evitar (ou diminuir) a resistência às mudanças antes de execução das próximas atividades do processo. Vale ressaltar que esta atividade deve abranger os membros de todos os projetos que serão migrados para GC.

A terceira atividade do processo é **diagnosticar o projeto**, cuja primeira tarefa é a realização de uma reunião entre os consultores e a equipe do projeto. A finalidade desta reunião é a equipe apresentar o projeto para os consultores, de modo a permitir que estes conheçam um pouco da estrutura e do histórico de desenvolvimento do mesmo. Além disso, a reunião permite aos consultores identificarem as características relevantes para a aplicação da GC no contexto do projeto, como: (1) desenvolvimento distribuído e concorrente, (2) número de desenvolvedores, (3) tamanho do projeto e (4) situação em relação à GC. Para que nenhum aspecto importante do projeto seja esquecido, um roteiro padrão é usado como referência. Após a reunião, a equipe de consultoria elabora o documento de diagnóstico, que representa o entendimento da equipe de consultoria sobre o projeto. Este documento também inclui uma sugestão inicial da estrutura do repositório de controle de versões, bem como a estratégia de implantação a ser utilizada. Após a elaboração, uma nova reunião é agendada para a apresentação deste diagnóstico, de maneira que a equipe de projeto avalie o documento e que os ajustes necessários sejam feitos.

As atividades (1) treinar equipe do projeto, (2) planejar a GC e (3) preparar a migração podem ser executadas em paralelo. Na atividade **treinar equipe do projeto**, a equipe do projeto que será migrado é preparada com o objetivo de assimilar as melhores práticas de GC, bem como se familiarizar com as ferramentas a serem utilizadas. Paralelamente, na atividade **planejar a GC**, o plano de gerência de configuração do projeto é criado. Este plano serve como um guia de referência para as ações relacionadas à GC que devem ser aplicadas ao longo do ciclo de vida do projeto. Este plano também facilita a integração de um novo membro na equipe de projeto, visto que as políticas adotadas para GC estão nele explicitadas. Para facilitar a criação do documento, é usado um modelo baseado na norma IEEE Std 828 [IEEE, 2005]. Este modelo pode ser adaptado de acordo com as particularidades do projeto. No final da atividade, o plano é apresentado para os envolvidos no projeto em uma reunião para avaliação.

Na atividade **preparar a migração**, são executados testes referentes à migração do projeto para os novos sistemas de GC, considerando uma nova estrutura de organização do repositório, caso necessário. Os testes servem como referência para a identificação de problemas referentes à migração e para que seja elaborada uma estratégia para a sua execução.

Depois que a equipe do projeto tiver sido treinada, o plano de controle de GC estiver aprovado e a estratégia de migração concluída, é realizada a atividade **migrar projeto**. Nesta atividade, os artefatos disponíveis são migrados para os respectivos sistemas de GC escolhidos para a organização.

Por fim, concluídas estas etapas, é realizada a atividade **realizar *mentoring***, na qual a equipe de consultoria esclarece quaisquer possíveis dúvidas da equipe de projeto

quanto à utilização dos procedimentos em seu cenário real. Esta etapa é fundamental para garantir a efetividade da implantação dos mecanismos de GC e para tornar a equipe de projeto apta a utilizar estes mecanismos com segurança e autonomia.

4. A Implantação de GC no CEPEL

Para avaliar o processo proposto, esta seção apresenta e discute sua utilização na implantação de controle de versões no CEPEL ao longo dos últimos três anos. O CEPEL é uma instituição que compõe o Sistema Eletrobras e possui mais de 30 anos em pesquisa e desenvolvimento relacionados à geração, transmissão e distribuição de energia elétrica. A maior parte dos profissionais atuantes na área de pesquisa desta instituição possui formação em Engenharia Elétrica.

O CEPEL possui vários projetos de software em desenvolvimento. Parte dos projetos não estava sob controle de versões. Na maioria das vezes, este controle era realizado pelo desenvolvedor, que normalmente organizava as versões anteriores sob a forma de diretórios em sistemas de arquivos, fazendo *backup* em mídias, como disquetes e CDs. Os projetos que já utilizavam um sistema de controle de versões (SCV) não o faziam de maneira uniforme. Alguns projetos utilizavam a ferramenta Visual Source Safe (VSS) [MICROSOFT, 2010a], outros o Concurrent Version System (CVS) [FREE SOFTWARE FOUNDATION, 2010] e um dos projetos utilizava o Subversion [COLLABNET, 2010a]. No entanto, alguns problemas foram identificados e relatados por diversos usuários (em especial com relação à ferramenta VSS, confirmados em um artigo da Microsoft [MICROSOFT, 2010b]), como, por exemplo, bases corrompidas e dificuldade da administração do controle de acesso. Esses problemas motivaram a criação de um projeto de parceria entre o CEPEL e a Universidade Federal do Rio de Janeiro com o objetivo de implantar GC no CEPEL.

O processo de implantação de GC proposto foi aplicado a 16 projetos do CEPEL, sendo dois projetos de grande porte, sete de médio porte e sete de pequeno porte[1]. Conforme pode ser observado na Tabela 1, os dois projetos de grande porte usavam um SCV. Dos sete projetos de médio porte, cinco usavam um SCV (71%). Por outro lado, nenhum dos sete projetos de pequeno porte usava um sistema de controle de versão. Esses dados indicam que os desenvolvedores de sistemas de grande e médio porte possuíam maior preocupação com a tarefa de controle de versão. Também é possível observar que o VSS era o principal SCV adotado.

Tabela 1. Descrição das características dos projetos.

Porte	Nenhum	SVN	CVS	VSS
Pequeno	7	0	0	0
Médio	2	1*	2*	3
Grande	0	0	0	2

*Um dos projetos usava SVN e CVS para subprojetos diferentes.

[1] Neste trabalho, entende-se por *projeto de pequeno porte* um projeto que possui menos de 100 arquivos e uma equipe com um ou dois desenvolvedores. Já um *projeto de grande porte* possui mais de 10.000 arquivos e uma equipe com mais de cinco desenvolvedores. Um *projeto de médio porte* encontra-se no intervalo entre os dois outros tipos de projeto mencionados.

Considerando que o foco inicial foi a implantação de um SCV padrão, o processo foi adaptado de forma a abranger apenas esta atividade. Desse modo, a preparação do ambiente envolveu a escolha e a implantação de software para controle de versões, e o plano de gerência de configuração derivou um plano de controle de versões.

4.1. Execução do Processo

Na atividade de preparação do ambiente, o Subversion foi escolhido como ferramenta para o controle de versões. Esta decisão foi baseada na função da robustez do produto, do grande número de usuários existentes e da variedade de ferramentas de suporte disponíveis. É importante notar que essa escolha ocorreu no início de 2007, quando os sistemas de controle de versão distribuídos ainda não apresentavam robustez e usabilidade suficientes para adoção em ambiente empresarial. Após a escolha e implantação do Subversion, foi realizada a atividade de conscientização para os membros de todos os projetos. Em seguida, foi iniciada a implantação do controle de versões para cada um dos 16 projetos selecionados.

A atividade de diagnóstico foi executada da mesma maneira para todos os projetos. Na atividade de treinamento, a equipe de cada projeto participou de um treinamento teórico sobre Subversion e um treinamento prático com as ferramentas a serem utilizadas: SVNManager [SOURCEFORGE, 2010], TortoiseSVN [COLLABNET, 2010b] e Subversive [POLARION, 2010]. Paralelamente ao treinamento, foi executada a atividade de preparação da migração. Os testes realizados nesta atividade ajudaram a detectar e resolver muitos problemas referentes aos repositórios baseados em outros sistemas de controle de versão, particularmente o VSS. Ao mesmo tempo, a atividade de planejamento de GC foi executada para a criação do plano de controle de versões.

A atividade seguinte foi a migração. Os artefatos do projeto foram migrados para o repositório Subversion do CEPEL. No caso dos projetos que não utilizavam nenhum SCV, a etapa de migração consistiu na inclusão dos artefatos do projeto no repositório criado para o mesmo. Em alguns casos, foi desejado incluir versões anteriores armazenadas em mídias, na forma de diretórios. Para isso, foi realizada a inclusão da versão inicial e, a cada nova versão a ser incluída, foi feita a comparação com a versão imediatamente anterior, de forma a armazenar apenas as diferenças entre as versões (como é comum ocorrer de forma automática com projetos que já se encontram sob controle de versão). Para cada versão que representava uma liberação (*release*), uma etiqueta (*tag*) foi criada. No caso dos projetos que utilizavam CVS ou VSS, essa etapa envolveu um processo de migração da respectiva base para Subversion. Finalmente, foi realizada a atividade de *mentoring*, a partir da qual as dúvidas dos membros de uma equipe eram resolvidas.

5. Considerações Finais e Lições Aprendidas

Considerando que a atividade de manutenção ganha destaque na indústria de software por exigir o maior esforço entre as atividades de Engenharia de Software, além dos altos custos financeiros, a GC pode contribuir para romper algumas das dificuldades relacionadas a esta atividade, tais como a falta de conhecimento sobre o produto e negligência na condução de mudanças no software. Nesse sentido, este artigo apresentou

um processo proposto para implantação de GC baseado na uniformização do uso de práticas e ferramentas e na preocupação com a cultura organizacional de desenvolvimento de software. A partir do processo proposto, verificou-se a sua aplicabilidade em uma organização que possui vários projetos (CEPEL), no que diz respeito ao controle de versões.

A partir da atividade de conscientização sobre os conceitos e a finalidade de GC no desenvolvimento de software, os *stakeholders* puderam entender a importância da implantação de um SCV para os seus projetos. As equipes de projetos de pequeno porte, que faziam o controle das versões manualmente, ainda não consideravam como um problema a falta de um ferramental de apoio, pois o tamanho da equipe e a quantidade de *releases* eram relativamente pequenos, e não havia trabalho concorrente. No entanto, graças ao treinamento e ao acompanhamento do processo, foi possível demonstrar que, à medida que o projeto cresce, o controle manual de versões pode se tornar bastante trabalhoso.

A duração das etapas de preparação e migração variou conforme o tamanho do histórico armazenado (e.g., em outro SCV, em pastas ou em CDs), o número de versões do projeto e o tipo de SCV usado. Os projetos que usavam VSS apresentaram maior dificuldade de migração, particularmente devido a problemas como a base do repositório estar corrompida.

O tempo médio para a execução desse processo foi de três meses para cada projeto. No entanto, esse tempo variou para mais ou para menos de acordo com alguns fatores, dentre eles a complexidade associada à reestruturação de repositórios (das equipes que já utilizavam outro SCV) e a disponibilidade da equipe do projeto para a realização das reuniões e para a etapa de migração.

O SCV adotado trabalha por padrão com a política otimista, ou seja, priorizando o trabalho concorrente. Os desenvolvedores que utilizavam VSS tiveram certa resistência quanto a essa característica, pois esse sistema utiliza, por padrão, a política pessimista (baseada em bloqueios). Esse fato foi contornado pela possibilidade de usar a mesma política no Subversion. Além disso, a escolha da política de controle de concorrência no Subversion pode ser realizada para cada artefato de forma independente. Finalmente, a resistência quanto ao uso das novas ferramentas, por parte de algumas equipes, foi contornada com a realização do treinamento e com a atividade de *mentoring*.

Também é importante notar que a inclusão de uma etapa de treinamento nos conceitos e práticas da área de GC muitas vezes eliminou resistências e dúvidas sobre a implantação. Portanto, o sucesso deste tipo de trabalho depende tanto da capacitação e conscientização dos envolvidos como do planejamento da nova estrutura de GC, além da correta implantação da mesma. Como trabalhos futuros, pretende-se analisar as estatísticas do repositório relacionadas aos projetos, executar um *survey* para capturar as percepções dos *stakeholders* frente a cada um dos projetos (tratando impactos técnicos e sociais da execução do processo proposto) e utilizar o processo proposto para tratar dos demais sistemas de GC.

Agradecimentos

Os autores agradecem a todos os profissionais do CEPEL que participaram deste projeto.

Referências

BHATT, P., SHROFF, G., MISRA, A.K. (2004) "Dynamics of Software Maintenance". *ACM SIGSOFT Software Engineering Notes*, v. 29, n. 5 (September), pp. 1-5.

CHRISTENSEN, M.J., THAYER, R.H. (2002) "*The Project Manager's Guide to Software Engineering's Best Practices*". IEEE Computer Society Press and John Wiley & Sons.

COLLABNET (2010a) "Subversion". Disponível em: <http://subversion.tigris.org/>. Acessado em 18/04/2010.

COLLABNET (2010b) "TortoiseSVN". Disponível em: <http://tortoisesvn.tigris.org/>. Acessado em 18/04/2010.

ESTUBLIER, J. (2000) "Software Configuration Management: A Roadmap", In: *Proceedings of the Conference on the Future of Software Engineering*, *22nd International Conference on Software Engineering*, Limerick, Ireland, pp. 279-289, June.

FREE SOFTWARE FOUNDATION (2010) "Concurrent Version System". Disponível em: <http://savannah.nongnu.org/projects/cvs/>. Acessado em 18/04/2010.

IEEE (2005) "*IEEE Std 828 – IEEE Standard for Software Configuration Management Plans*". Institute of Electrical and Electronics Engineers.

KAJKO-MATTSSON, M., FORSSANDER, S., OLSSON, U. (2001) "Corrective Maintenance Maturity Model (CM^3): Maintainer's Education and Training", In: *Proceedings of the 23rd International Conference on Software Engineering*, Toronto, Canada, pp. 610-619, May.

LEON, A. (2000) "*A Guide to Software Configuration Management*". Artech House Publishers.

MCT (2006) "*Qualidade e Produtividade no Setor de Software Brasileiro*". Ministério de Ciência e Tecnologia, Secretaria de Política de Informática.

MICROSOFT (2010a) "Visual Source Safe". Disponível em: <http://msdn.microsoft.com/pt-br/library/ms181038(VS.80).aspx>. Acessado em 18/04/2010.

MICROSOFT (2010b) "Como detectar e corrigir erros de corrupção de banco de dados no Visual SourceSafe para Windows 6.0 e no SourceSafe". Disponível em <http://support.microsoft.com/kb/133054>. Acessado em 21/04/2010.

MURTA, L.G.P. (2006) "*Gerência de Configuração no Desenvolvimento Baseado em Componentes*". Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

PADUELLI, M.M., SANCHES, R. (2006) "Problemas em Manutenção de Software: Caracterização e Evolução", In: *Anais do III Workshop de Manutenção de Software Moderna*, *V Simpósio Brasileiro de Qualidade de Software*, Vila Velha, Brasil, pp. 1-13, Junho.

POLARION (2010) "Subversive". Disponível em: <http://www.polarion.org/index.php?page=overview&project=subversive/>. Acessado em 18/04/2010.

POLO, M., PIATTINI, M., RUIZ, F., CALERO, C. (1999) "Roles in the Maintenance Process". *ACM SIGSOFT Software Engineering Notes*, v. 24, n. 4 (July), pp. 84-86.

SCHACH, S.R., TOMER, A. (2000) "A Maintenance-Oriented Approach to Software Construction". *Journal of Software Maintenance: Research and Practice*, v. 12, n. 1 (January-February), pp. 25-45.

SOFTEX (2009) "Guia *Geral do MPS.BR – Modelo MPS e Modelo de Referência (MR-MPS)*". Sociedade SOFTEX, Maio de 2009.

SOURCEFORGE (2010) "SVNManager". Disponível em: <http://svnmanager.sourceforge.net/> Acessado em 18/04/2010.