Requisitos de Ferramentas de Gerenciamento de Configuração

Viviane Nogueira Pinto de Oliveira[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627 –CEP 31.270-010 – Belo Horizonte – MG – Brasil

vivianen@dcc.ufmg.br

Abstract. *This paper presents requirements of configuration management tools, guiding the choice and effective use of these tools in software development. This article introduces the configuration management discipline, then presents the requirements and description of practical and operational aspects.*

Resumo. *Este trabalho apresenta requisitos de ferramentas de gerenciamento de configuração, guiando a escolha e uso efetivo de tais ferramentas pela comunidade envolvida no desenvolvimento de software. Este artigo contextualiza a disciplina de gerenciamento de configuração, seguindo-se com a elucidação dos requisitos e descrição de aspectos práticos e operacionais.*

1. Introdução

No processo de desenvolvimento de software mudanças ocorrem a todo momento. As mudanças ocorrem por inúmeras razões, tais como, novas condições de negócios e de mercado, novas necessidades de clientes, reorganização, crescimento ou declínio dos negócios, restrições de tempo e custo. Tudo isso pode demandar mudanças nas prioridades do software, na estrutura do time de desenvolvimento, nas informações produzidas pelo software, nos requisitos de produto e nas regras de negócio [Pressman 2001, pg 254].

Neste contexto, surge o gerenciamento de configuração de software (GCS) como uma disciplina da engenharia de software que é responsável pelo desenvolvimento da gerência da mudança durante todo o ciclo de vida do software [Pressman 2001, pg 255]. O GCS é uma das ramificações mais sucedidas da engenharia de software, tanto que é atualmente considerado uma ferramenta essencial para o sucesso de qualquer projeto de desenvolvimento de software, sendo necessário para o atendimento do segundo nível de maturidade do CMM [Estublier 2002, pg 5]. Facilidades para acomodar mudanças, maior controle sobre os produtos e economia de tempo de desenvolvimento são alguns dos benefícios da adoção da GCS, no entanto é importante considerar também os custos associados, tais como treinamento e recursos computacionais [ITI 2001, pg 22].

O GCS engloba as atividades de identificação da mudança, controle da mudança, garantia de efetividade da implementação da mudança e divulgação das mudanças para outros interessados [Pressman 2001, pg 253].

Para controlar as mudanças é preciso controlar os itens gerados pelo processo de desenvolvimento, assim considerados documentos de especificação e desenho, materiais

de testes, ferramentas de software, códigos-fonte e executáveis, bibliotecas de código, dicionário de dados, documentação para instalação, manutenção, operação e uso do software [SWEBOK 2004, pg 111].

Diante da complexidade de itens e atividades, surge a necessidade de um processo automatizado de controle de mudanças, que ofereça tarefas de suporte ao gerenciamento de configuração. O controle efetivo se traduz em qualidade de software [Pressman 2001, pg 260]. As ferramentas de gerenciamento de configuração de software são alternativas para a abstração da complexidade do ambiente de desenvolvimento de software [Schamp 1995].

Este trabalho tem o objetivo de apresentar um catálogo com os requisitos de ferramentas de gerenciamento de configuração, tanto para a aquisição de uma nova ferramenta, quanto o aperfeiçoamento do uso das ferramentas já utilizadas. Existem diversas ferramentas de GCS disponíveis no mercado, sendo que a maioria possui foco no controle de versões e controle de liberações de arquivos-fonte, negligenciando outros requisitos importantes do GCS; um exemplo é a atividade de auditagem de configuração que tem recebido pouca atenção [Chan, Hung 1997], o qual é um aspecto a ser tratado neste trabalho. A comunidade envolvida em processos de desenvolvimento de software em geral, tais como especificadores, implementadores, testadores, arquitetos de software e etc, são os principais interessados neste trabalho, no qual serão estabelecidas necessidades, restrições e expectativas em relação as ferramentas de gerenciamento de configuração.

Para este trabalho, pretende-se inicialmente realizar uma pesquisa sobre gerenciamento de configuração, identificando os conceitos básicos, por meio do livro de Pressman [Pressman 2001], o guia do corpo de conhecimento da engenharia de software [SWEBOK 2004], o guia de gerenciamento de configuração da organização IEEE [IEEE 1987], além de outros. Os requisitos do catálogo serão especificados seguindo um estudo bibliográfico principalmente dos trabalhos de Estublier [Estublier 2002], Wang e Ren [Wang, Ren 2006], Chan e Hung [Chan, Hung 1997], Schamp [Schamp 1995], e Grinter [Grinter 1995] relacionados nas referências bibliográficas, além daqueles cujo suporte serão importantes para o desenvolvimento deste artigo.

2. Modelos e características de Gerenciamento de Configuração

A construção de um software é uma tarefa complexa, por isso processos de software são necessários para a abstração da complexidade. Os processos estão relacionados à construção do software propriamente dito e àqueles que visam o controle da qualidade. O processo de gerenciamento de configuração de software é um processo que objetiva a qualidade, controlando as informações geradas por outros processos [Cunha, Prado, Santos 2004].

O gerenciamento de configuração surgiu da necessidade de se controlar mudanças ocorridas durante todo o processo de desenvolvimento de software. O GCS controla a evolução e integridade do produto pela identificação de seus elementos, gerenciando e controlando mudança, verificando, gravando e reportando informação de configuração.

As atividades de gerenciamento de configuração de software são identificação, controle, administração do estado, auditagem e gerenciamento de liberações e entrega da configuração do software. Estas atividades devem ser planejadas inicialmente, provendo assim um plano a ser seguido durante todo o processo de GCS [SWEBOK 2004, pg

108].

2.1. Planejamento do GCS [IEEE 1987, pg 17-23]

Um efetivo processo GCS envolve planejamento de atividades a serem executadas, o qual é essencial para o seu sucesso.

O planejamento da GCS deve ser consistente com o contexto organizacional, assim considerando as restrições e a natureza do projeto. No planejamento são identificados responsabilidades, ferramentas, cronogramas, marcos dos projetos, requisitos de treinamento. Também são planejados a seleção e implantação de ferramentas, subcontratação, estabelecimento de interfaces e dependências.

Ao final do planejamento é gerado um documento que servirá de base para todo o processo de GCS, chamado Plano de Gerenciamento de Configuração de Software, contendo escopo, propósito, definições e referências.

2.2. Identificação dos Itens

O processo de desenvolvimento de software gera inúmeras saídas, que devem ser identificados para o estabelecimento da configuração inicial do software. Exemplos de itens de configuração, também denominados entidades, são [IEEE 1987, pg 10]:

- Planos de gerenciamento;
- Especificações de requisitos e desenho;
- Documentação de usuário;
- Projetos de teste, casos de teste;
- Software de suporte;
- Dicionários de dados;
- Código-fonte;
- Código executável;
- Bibliotecas;
- Bases de dados com dados processados ou aqueles que são parte de um programa;
- Documentação de manutenção.

As ferramentas utilizadas na construção do software também são itens de configuração, pois elas são usadas para produzir artefatos do processo de desenvolvimento de software. Elas precisam ser controladas, pois também podem ocorrer mudanças em suas versões [Pressman 2001, pg 229].

Na fase de identificação da configuração de software estabelece-se as linhas de base, permitindo muitas pessoas trabalharem ao mesmo tempo. Define-se também os responsáveis pelas criações, aprovações e manutenções de linhas de base [IEEE 1987, pg 24].

Nesta atividade a capacidade de estabelecer relações entre itens é vital, pois garante ao desenvolvedor que as mudanças ocorridas em itens de origem se reflitam em mudanças relevantes na documentação [Chan, Hun 1997].

2.3. Controle

Uma mudança normalmente se inicia com uma requisição. Uma pessoa ou um time avalia a requisição, aprovando ou não a mudança. Se a mudança for aprovada, ela é

implementada, testada e então solucionada [Estublier 2002, pg 13].

A atividade de controle objetiva gerenciar mudanças durante o ciclo de vida do software, provendo a identificação de procedimentos a serem usados para processar mudanças nas linhas de base, verificação de níveis de autoridade para realizar mudanças especificados no plano, verificação dos tipos das mudanças dentre outras tarefas [IEEE 1987, pg 26].

A atividade de controle é fortemente facilitada pelo uso de ferramentas que proporcionam benefícios, por exemplo, rastreamento de soluções para problemas reportados durante o processo de requisição de mudança e versionamento dos códigos-fonte no processo de implementação da mudança.

2.4. Administração de Estado

Esta atividade visa o controle do estado da configuração do software, por meio da coleta de informações e a geração de relatórios. Esta atividade pode ser comparada com um sistema de contas, no qual muitos dos conceitos usados para rastrear o fluxo de fundos através das contas podem ser usados para rastrear o fluxo de software através de sua evolução. O objetivo desta atividade é basicamente reportar transações que ocorrem entre entidades controladas pelo GCS [IEEE 1987, pg 30].

Além de reportar informações do estado corrente da configuração, a informação obtida pela atividade de administração de estado pode servir como base de várias medições de interesse ao gerenciamento, desenvolvimento e GCS, por exemplo, o número de requisições de mudanças por item de configuração e a média de tempo necessária para implementar a mudança [SWEBOK 2004, pg 113].

2.5. Auditagem

Esta atividade está relacionada à verificação da conformidade dos produtos e processos de software aos padrões, guias, planos e procedimentos estabelecidos [SWEBOK 2004, pg 114]. Dentre outras funções, verifica-se se itens de configuração identificados estão presentes na linha de base e se cada um deles satisfaz as funções definidas na especificação ou contrato para o qual foi desenvolvido [IEEE 1987, pg 32].

O processo de desenvolvimento de software deve ser verificado, validado por meio de revisões formais e inspeções executadas em arquivos de código e documentos [Chan, Hung 1997].

2.6. Gerenciamento e entrega de liberações

O processo de integração do software (*building software*) e gerenciamento de liberações são funções executadas nesta atividade do GCS. O processo de integração do software provê a combinação de versões de itens de configuração, usando dados apropriados de configuração em um programa executável para entrega ao cliente ou a outra atividade do processo de desenvolvimento, tal como a atividade de teste [SWEBOK 2004, pg 114]. Este processo executa passos numa seqüência definida, permitindo assim reproduzir um executável de uma versão anterior. Exemplos de ferramentas usadas neste processo são os compiladores [SWEBOK 2004, pg 114].

A atividade de gerenciamento de liberações empacota o produto da atividade de integração de software juntamente com outros produtos do processo de desenvolvimento, como a documentação do software e instruções de instalação e atualização [SWEBOK 2004, pg 115].

3. Requisitos de Ferramentas

Para o controle efetivo e apoio às atividades relacionadas à mudança, o uso de ferramentas é de suma importância. Existem ferramentas de GCS específicas para cada tipo de atividade, por exemplo, para um efetivo processo de requisição de mudança de software é importante o uso de ferramentas de suporte para o recebimento de requisições e o fluxo de processo de mudança, capturando decisões gerenciais e reportando informações. Já na implementação da mudança as ferramentas utilizadas são aquelas de suporte ao controle de versão e suporte ao armazenamento de código [SWEBOK 2004, pg 113].

O processo de seleção de uma ferramenta faz parte do processo de planejamento visto na seção 2.1, no qual diversos fatores devem ser levados em consideração [Schamp 1995].

Nesta seção será mostrado um conjunto de requisitos de ferramentas de configuração, em termos de capacidades a serem oferecidas. Esta seção tem o objetivo de guiar a escolha de uma ferramenta apropriada ao ambiente organizacional no qual o time de desenvolvimento está inserido.

3.1. 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, pg 233].

Controle de versões é usado no suporte à proliferação de itens gerados no processo de desenvolvimento de software, provendo o gerenciamento deles, gravando seus relacionamentos e suas propriedades comuns [Estublier 2002, pg 3]. 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. Exemplos de objetos foram citados na seção 2.2.

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 [Chang, 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 [Chang, Hung 1997].

Para o estabelecimento do 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: 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 [Chan, Hung 1997].
- *Check in*/*check out*: Antes de começar a trabalhar num 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 colegas. 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 2002, pg 7]. 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 [Schamp 1995].
- Controle de liberações: Identifica versões marco do sistema. Esta característica numa ferramenta é importante para equipes de desenvolvimento com claras definições e padrões bem definidos para mover um sistema através do ciclo de vida do software [Schamp 1995]. O controle de liberações provê um mecanismo de conexão das requisições de mudança com os conteúdos das liberações [SWEBOK 2004, pg 115].
- 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 e não apenas textuais [Schamp 1995].
- Trilha de auditagem: As atividades executadas em itens de configuração sobre o controle de verões devem ser gravadas [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 requer 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. Um outro exemplo é que mudanças ocorridas em versões rascunho de documentos são menos controladas que mudanças nas versões finais [IEEE 1987, pg 26].
- Suporte ao repositório de código [SWEBOK 2004, pg 113]:
- 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.

3.2. Integração de Software (*Building*)

Este requisito automatiza as tarefas relacionadas com a compilação do software de forma a produzir um sistema de software e geração de arquivos executáveis para colocá-lo em produção ou para outra destinação [Schamp 1995]. Além disso, este requisito na ferramenta provê a redução ou eliminação de erros humanos [Chan, Hung 1997]. Estublier [Estublier 2002, pg 12] afirma que informações que tem que ser providas e mantidas de maneira manual representam um alto custo.

3.3. Interfaces Padronizadas

As ferramentas usadas no suporte ao gerenciamento de configuração devem possuir interfaces com outras ferramentas usadas no processo de software, tais como, ferramentas de análise, desenho, construção e documentação [Schamp 1995].

E além de apenas interfaces, padrões são importantes para permitir que se uma ferramenta não satisfizer todos os requisitos desejados pelos usuários, pelo menos ela será capaz de integrar outras ferramentas que possuem tal característica [Schamp 1995].

4. Aspectos Práticos e Operacionais

Conforme mencionado na seção 3, diversos são os fatores que influenciam a seleção de uma ferramenta de GCS. As características do projeto guiam a seleção da ferramenta a ser utilizada [IEEE 1987, pg 15]. A medida que o projeto cresce em seu tamanho também cresce a dificuldade para estabelecer o suporte de uma ferramenta automatizada.

No entanto, o suporte de ferramentas que satisfaça a implementação planejada é importante, mas uma condição insuficiente para o sucesso geral. Alcançar todos os benefícios do GCS requer disciplina e dedicação de todos os níveis de gerenciamento e de pessoas da organização, desde a concepção do sistema até a sua liberação final. Custos de aquisições das ferramentas e treinamento devem ser considerados [Schamp 1995].

Embora existam ferramentas de automatização dos processos, elas não possuem suporte suficiente a algumas das atividades básicas preconizadas na teoria do GCS, entre as quais, a atividade de identificação de configuração, mecanismos de auditagem e geração relatórios e atividades de medição [Wang, Ren 2006].

5. Conclusão

Apresentou-se neste trabalho uma lista de requisitos que guiam a escolha de ferramentas de GCS, assim entendidos também as restrições e observações a serem adotadas na sua seleção. Para a escolha da ferramenta é necessário o planejamento inicial, no qual fatores, como o contexto no qual a organização está inserida, devem ser verificados para a conformidade com os objetivos estabelecidos.

Referências

[Pressman 2001] Pressman, Roger S. *Software Engineering - A practitioner's Approach*. – 5ª edição, McGraw-Hill, 2001.

[SWEBOK 2004] *Guide to the Software Engineering Body of Knowlegment,* 2004. Disponível em http://www.swebok.org/htmlformat.html. Acesso em 21/08/2007.

[IEEE 1987] IEEE - The Institute of Electrical and Electronics Engineers, *IEEE Guide to Software Configuration Management*, ANSI/IEEE std 1042-1987, 1987

[Estublier 2002] Estublier, J.; D. Leblang, G.; Clemm, R.; Conradi, W.; Tichy, A. van der Hoek, and D. Wiborg-Weber, *Impact of the Research Community on the Field of Software Configuration Management*, ACM Software Engineering Notes, vol. 27, no. 5, 2002.

[Wang, Ren 2006] Wang, F.; Ren, A. *A Configuration Management Supporting System Based on CMMI*, 2006.

[Chan, Hung 1997] Chan, A. K. F.; Hung, S. *Software Configuration Management Tools*, 1997.

[Schamp 1995] Schamp, A. *CM-Tool Evaluation and Selection*. CASE Associations, 1995.

[Cunha, Prado, Santos 2004] Cunha, J. R. D. D. C.; Prado, A. F.; Santos, A. C. dos. *Uma Abordagem para o Processo de Gerenciamento de Configuração de Software*, Revista Eletrônica de Sistemas de Informação (RESI), São Paulo, v. III, 01 nov. 2004.

[Cunha 2005] Cunha, J. R. D. D. C., *SoCManager: Uma Ferramenta de Apoio ao Gerenciamento de Configuração de Software*, São Paulo, 2005.

[Grinter 1995] Grinter, R. E. *Using a Configuration Management Tool to Coordinate Software Development*, 1995.

[ITI 2001] Oliveira, A. A. C. P.; Primo, F. F.; Cruz, J. L; Martino, W. R. *Gerência de Configuração de Software: Evolução de Software sob Controle.* ITI – Instituto Nacional de Tecnologia da Informação, São Paulo, 2001.