Um Estudo sobre Gerência de Configuração de Software aplicada ao Desenvolvimento Baseado em Componentes

Cristine Ribeiro Dantas
Hamilton Oliveira
Leonardo Gresta Paulino Murta
Cláudia Maria Lima Werner
COPPE/UFRJ – Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
Caixa Postal 68511 – CEP. 21945-970
Rio de Janeiro – Brasil
{cristine, hamilton, murta, werner}@cos.ufrj.br

Abstract

This study aims to analyze the Software Configuration Management (SCM) approach for conventional development and identify points that need to be tailored to support Component Based Development (CBD). We believe that adapting SCM process to support CBD by applying variants for producer and consumer teams is the best way to take advantage of the benefits of SCM adoption, such as faults reduction and productivity increase. This work focuses on issues related to CBD that are not seen on conventional SCM approaches, and shows a brief survey of the current research on this field of joining DBC and SCM.

Resumo

O objetivo deste estudo é analisar a abordagem proposta pela disciplina de Gerência de Configuração de Software (GCS) para o desenvolvimento convencional e identificar os pontos que necessitam ser adaptados para o Desenvolvimento Baseado em Componentes (DBC). Acreditamos que a adaptação do processo de GCS, com variantes para equipes produtoras e consumidoras de componentes reutilizáveis, seja a alternativa mais viável para obter, no contexto de DBC, os resultados prometidos pela GCS, como, por exemplo, diminuição de falhas e aumento da produtividade. Este trabalho tem como foco principal o levantamento de questões referentes a DBC atualmente negligenciadas pela abordagem convencional de GCS, e mostra os direcionamentos de alguns autores para os atuais problemas encontrados no DBC.

Palavras-chave: Gerência de Configuração de Software, Desenvolvimento Baseado em Componentes, Controle de Versões.

1 Introdução

O Desenvolvimento Baseado em Componentes (DBC) é um paradigma de desenvolvimento que emergiu no contexto de reutilização de software. Alguns pesquisadores ressaltam que as questões referentes à evolução e adaptação de componentes impactam sua efetiva utilização (BRAGA, 2000). Além disso, BRERETON (1999) argumenta que a dificuldade é grande, principalmente devido à distribuição da responsabilidade de evolução entre diferentes desenvolvedores e organizações proprietárias de componentes. E, ainda, LARSSON (2000) enfatiza que o uso de dependências entre componentes utilizando suas interfaces, prática comum em DBC, colabora para que se possa pensar na necessidade de uma gerência de configuração de componentes.

A Gerência de Configuração de Software (GCS) é uma abordagem disciplinada para gerenciar o processo de evolução do desenvolvimento e manutenção do software (BURROWS, 1996). Vários autores propõem a utilização de ferramentas de GCS para resolver alguns problemas e questões que surgiram com DBC, tais como (CRNKOVIC et al., 2000; LEBSACK et al., 2001; LU ZHANG et al., 2001): (1) a manutenção da consistência na integração de componentes para a formação de um produto, (2) a incorporação de novas funções por parte dos desenvolvedores quando se faz uso de componentes de prateleira

(COTS), e (3) o relacionamento de um pedido de mudança de um produto em relação a outros produtos que utilizam os mesmos componentes.

Com as novas tecnologias de DBC, torna-se necessária a reengenharia de componentes legados. Para que essa reengenharia se torne viável e possibilite a obtenção das funcionalidades originais no novo componente, devem ser aplicadas não somente técnicas específicas para esse fim, como também um processo de GCS (CRNKOVIC et al., 2000). Além disso, a manutenção de software, que é uma atividade inevitável, chega a consumir 75% do custo total de ciclo de vida (BLANCHARD, 1991). Por esse motivo, são necessários mecanismos eficazes que possibilitem o controle da mudança, pois ambientes de desenvolvimento que não controlam mudanças são levados rapidamente ao caos (PRESSMAN, 2000).

A partir desta motivação, o objetivo deste trabalho é analisar a abordagem proposta por GCS para o desenvolvimento convencional, e identificar os pontos que devem sofrer modificação, em função das necessidades detectadas, para adequá-la ao desenvolvimento de software baseado em reutilização, especificamente ao DBC. Como GCS é uma área mais antiga e consolidada que DBC, o caminho mais indicado aparenta ser partir dessa infra-estrutura já existente de GCS para desenvolvimento convencional e adaptar características referentes ao DBC.

Além dessa primeira seção, que apresenta a motivação e o objetivo do trabalho, o artigo é composto por mais três seções. A segunda apresenta uma breve introdução sobre GCS. A terceira descreve o processo de GCS no contexto de DBC, produzido com base no estudo de GCS convencional e desenvolvimento *de* e *com* componentes. Finalmente, a quarta seção apresenta a conclusão do trabalho, exibindo o estado da arte no tema e descrevendo as contribuições e trabalhos futuros.

2 Gerência de Configuração de Software

A GCS pode ser definida como uma disciplina que identifica a configuração de um sistema em pontos discretos no tempo, com o objetivo de controlar sistematicamente as mudanças dessas configurações e manter a integridade e a rastreabilidade do ciclo de vida do sistema (LEON, 2000). Segundo a IEEE Std. 828 e ISO 10007, a GCS é composta por quatro funções principais: (1) identificação, (2) controle, (3) acompanhamento e (4) auditoria.

A função de identificação é a necessária para dar início ao controle sobre os itens de configuração (IC) selecionados. Um IC pode ser qualquer artefato que demanda GCS. Nesta função são executadas tarefas referentes à seleção e documentação dos ICs, que podem estar em diferentes níveis de abstração (análise, projeto, codificação, teste, etc).

A função de controle é responsável pela autorização, implementação e verificação das modificações sobre os ICs. As tarefas que compõem a função de controle são: (1) requisição de modificação, (2) classificação, (3) análise, (4) avaliação, (5) implementação e (6) verificação. Após a execução completa de um ciclo de controle, deve ser definida uma nova *baseline*[1] do sistema. Posteriormente ao estabelecimento de uma *baseline*, os ICs só podem ser alterados utilizando esse processo formal de controle da mudança.

A função de acompanhamento visa armazenar todas as informações produzidas nas demais funções e relatar essas informações aos desenvolvedores interessados e autorizados.

A função de auditoria é aplicada no momento em que se deseja gerar *releases*[2]. O seu objetivo é verificar a corretude da *release*, examinando os planos, dados e resultados de teste,

[1] Uma *baseline* agrega ICs que funcionam corretamente em conjunto e serve como base para a próxima etapa de desenvolvimento.

[2] Uma *release* é uma versão de *baseline* a ser entregue para o cliente.

e a completude da *release*, averiguando se os artefatos definidos contratualmente no início do projeto estão sendo entregues.

Os sistemas de GCS fazem uso de dois tipos principais de sub-sistemas, que são sistemas de controle de versões e sistemas de controle de modificações. Os sistemas de controle de versões visam manter o histórico das alterações efetuadas sobre cada IC existente. Já os sistemas de controle de modificações visam possibilitar a requisição e o acompanhamento das modificações no sistema.

Como o foco principal desse trabalho se situa na função de controle, essa função será descrita em mais detalhes. Inicialmente, uma requisição de modificação (CR – *change request*) é criada utilizando um sistema de controle de modificações. A CR é classificada de acordo com seu grau de relevância. Após a classificação, a CR passa por um processo de análise de impacto, que visa detectar o custo e o cronograma relacionados à modificação. A partir do próprio CR e da sua análise, o *Change Control Board*, comitê responsável por avaliar os CRs, avalia qual será o destino do CR em questão. Se o CR for aprovado, dar-se-á o início da sua implementação, utilizando um sistema de controle de versões. Caso contrário, deverá ser explicitado ao solicitante o motivo da recusa. Após a implementação, deverá ocorrer uma verificação sobre o que foi implementado para assegurar a consistência da nova *baseline*.

3 Considerações sobre Processo de GCS Convencional para Apoiar DBC

O processo de DBC é dividido em duas etapas: desenvolvimento *de* componentes (produtor) e desenvolvimento *com* componentes (consumidor). BRAGA (2000) ressalta que o conceito de componentes deve ser mais do que somente código, enfatizando uma abordagem de desenvolvimento de componentes como um processo de engenharia de domínio completo, onde a preocupação está voltada para uma família de aplicações. Esse contexto potencializa os problemas já existentes no desenvolvimento convencional em relação à evolução de artefatos. Por esta razão, a utilização de GCS é fundamental para viabilizar o sucesso de DBC.

No processo de GCS, foram analisadas as funções de Identificação da Configuração e Controle da Configuração, com o intuito de adaptá-las a este novo contexto de desenvolvimento.

3.1 Adaptação da função de identificação

Em função da demanda dos consumidores, pode ser priorizada uma ordem de construção de novos componentes. Neste contexto, os itens de configuração podem ser classificados como internos, quando são desenvolvidos pela própria equipe consumidora, ou externos, quando são vindos da equipe produtora, seja ela pertencente à própria organização ou de terceiros.

A documentação dos itens de configuração pode variar em relação a esta classificação. Para cada participante da equipe, pode ser definido um nível de acesso que determine a visibilidade da informação. O que, no entanto, deve ser ressaltado é que toda a documentação que estiver disponível para o item de configuração, inclusive a descrição de algoritmos e código fonte, deve ser adicionada ao item, para que o mesmo possa ser projetado e testado de forma independente.

A granularidade do item de configuração é outro ponto questionável. Podemos assumir heurísticas para estipular o que considerar como item de configuração, visto que este pode englobar qualquer artefato produzido durante o desenvolvimento de software, ou um conjunto de artefatos. Uma heurística seria considerar elementos coesos e fracamente acoplados.

Deve-se ainda considerar a possibilidade de definição da interface como item de configuração em DBC. Apesar de se assumir a estabilidade dos contratos de interface como

premissa para a compatibilidade entre componentes, em certas circunstâncias até as próprias interfaces evoluem, e todos os clientes de componentes que requerem ou provêem essas interfaces devem ser notificados para que possam se adequar, de forma não traumática, a essas modificações contratuais.

3.2 Adaptação da função de controle

O controle da configuração seria alterado tanto na equipe produtora quanto nas equipes consumidoras para contemplar as seguintes considerações em cada passo do processo.

3.2.1 Adaptação da função de controle na equipe produtora

A requisição da modificação partiria dos grupos consumidores de componentes. Na classificação seria necessário levar em conta o impacto da modificação dentre os diversos grupos consumidores. Quanto à análise, se a modificação implicasse na criação de novos componentes, seria necessário avaliar, além de custo e tempo de desenvolvimento, o interesse dos demais grupos consumidores. A avaliação, a implementação e a verificação não necessitariam de adaptações em relação ao processo tradicional de GCS. Contudo, na função de acompanhamento seria necessária a notificação a todos os envolvidos sobre a modificação. Para que esta ação fosse possível, o produtor deveria manter o rastro do componente em relação a todas as equipes consumidoras que adquiriram o componente, assim como um controle das versões utilizadas por cada uma dessas equipes.

3.2.2 Adaptação da função de controle nas equipes consumidoras

Tanto a requisição da modificação quanto a classificação não sofreriam adaptações em relação ao processo tradicional. Já na análise, caso a modificação fosse em um componente reutilizável, seria necessário interagir com o processo de GCS da equipe produtora de componentes para que fosse possível estimar e avaliar o impacto da mudança. Ainda na análise, deveria ser avaliado se usar um componente de outro fornecedor seria mais econômico, ao invés de modificar o existente, desde que esse novo componente obedecesse a especificação desejada. Se o componente fosse um COTS, o componente somente poderia ser modificado pelo fornecedor. Caso o componente fosse do tipo caixa branca, o consumidor poderia avaliar se é mais econômico e viável fazer a modificação dentro do seu próprio processo de desenvolvimento (*in-house*), ao invés de fazer uma requisição de mudança no fornecedor. Para componentes caixa preta, também existe a possibilidade da definição de adaptadores (GAMMA, 1994).

Na avaliação, a opção de aceitar a modificação passa a ser dividida entre aceitar optando pelo desenvolvimento ou optando pela reutilização de componentes existentes. Neste último caso, é criada uma nova opção de escolha além das existentes no processo de GCS. Existirá uma requisição de modificação no processo de GCS da equipe produtora caso seja aprovada, no consumidor, a substituição do componente atual por outra versão.

Caso a implementação seja feita na equipe produtora, a equipe consumidora deve, ao receber a nova versão do componente, empacotá-lo em um item de configuração dentro de seu processo de GCS e documentá-lo para que fique identificado o produtor do componente, as restrições e os direitos contratuais envolvidos na aquisição.

O consumidor deve verificar o componente realizando testes de integração e regressão. O consumidor não pode excluir os testes mesmo depois de o produtor assegurar que o componente, em conformidade com as suas interfaces, atende à especificação anterior ou especificação acrescida de uma nova funcionalidade. Os testes de regressão e integração também são importantes como forma de averiguar se os requisitos não funcionais continuam sendo atendidos no contexto de uso do componente (SILVA et al., 2003).

4 Conclusão

O objetivo deste trabalho foi analisar a abordagem proposta pela disciplina de GCS para o desenvolvimento convencional, e identificar os pontos que devem ser adaptados para adequá-la ao desenvolvimento de software baseado em componentes. Esses pontos foram identificados a partir das dificuldades encontradas no processo de DBC, que poderiam ser contornadas com o uso de GCS. Ao longo do texto, foram descritos alguns aspectos que merecem melhor reflexão no contexto de DBC, por introduzirem preocupações que não existiam em ambientes convencionais.

Alguns trabalhos estão sendo feitos para apoiar a aplicação de GCS em DBC. Contudo, a atenção sobre esse tema ainda está aquém da sua importância no DBC. Dentre esses trabalhos, LU ZHANG et al. (2001) propõem o protótipo de um sistema de GCS, para atuar no processo de DBC. Esse protótipo deixa a desejar em questões como notificação de modificações, apoio a testes de integração, papéis do processo de aprovação de modificações e controle das interfaces implementadas pelos componentes. Segundo os autores, existe a ausência de modelos e técnicas que apóiem a gerência de versões de componentes, o que motivou o foco do trabalho em controle de versões, ignorando questões mais amplas de GCS.

Outro trabalho, apresentado por LEBSACK et al. (2001), aborda questões referentes ao controle dos itens de configuração em DBC, com foco em COTS. O protótipo de uma ferramenta é apresentado. A motivação deste trabalho está relacionada à dificuldade de identificar a disponibilidade de componentes e ao esforço para a construção de uma biblioteca capaz de armazená-los e organizá-los. Foi detectada a necessidade de controles mais sofisticados que os convencionais para que se possa aplicar GCS em COTS.

Para que se possa fornecer ferramental e processo de GCS adequados para DBC, é necessário um maior investimento em pesquisa, pois o estado da arte ainda é imaturo. Contudo, não é necessário iniciar do zero essas pesquisas. A possibilidade de adaptar as atuais técnicas de GCS de sistemas convencionais para DBC atenua o esforço necessário. Seguindo esse ponto de vista, estão sendo disponibilizados alguns serviços de GCS para o Ambiente Odyssey (MURTA et al., 2003), dentre eles:

- Adaptação do processo de GCS convencional para utilização em um ambiente de reutilização (iniciado neste trabalho) e posterior automatização;
- Controle de evolução dos contratos de interface dos componentes, notificando evolução aos consumidores que dependem de determinada interface;
- Identificação, busca e localização dos itens de configuração, possibilitando o acesso a diferentes versões de um mesmo item;
- Tributação sobre o desenvolvimento e o uso de componentes, possibilitando políticas de cobrança e bonificação;
- Apoio na seleção de componentes em função das características desejáveis para uma determinada configuração;
- Nomeação dos cortes ocorridos durante a seleção fazendo uso de rótulos, que possibilitam a reutilização dos próprios cortes de seleção;
- Aviso às equipes consumidoras sobre a evolução de componentes reutilizados por elas;
- Visualização gráfica do grau de modificações sofridas em um componente entre versões;
- Apoio na evolução das adaptações implementadas sobre componentes que necessitaram sofrer modificações;
- Controle de versões de componentes em alto nível de abstração, rompendo a metáfora de arquivo de sistema operacional.

Atualmente, estamos realizando pesquisas referentes à evolução de componentes dentro de ambientes de reutilização, onde as modificações requisitadas em uma aplicação podem implicar em modificações que afetam todo um domínio de aplicações (OLIVEIRA, 2003). Além disso, também estamos realizando pesquisas sobre a rastreabilidade de especificações de componentes a partir da coleta de informações oriundas da aplicação de GCS (DANTAS, 2003). Como trabalho futuro está prevista a implementação dessas propostas em protótipos que possibilitem a avaliação da sua aplicabilidade.

5 Referências Bibliográficas

ANSI/IEEE Std-828, 1998, "IEEE Standard for Software Configuration Management Plans", disponível em http://standards.ieee.org/reading/ieee/std_public/description/se/828-1990_desc.html, acessado em 16/06/2003.

BLANCHARD, B.S., 1991, System Engineering Management, New York: John Wiley & Sons.

BRAGA, R., 2000, *Busca e Recuperação de Componentes em Ambiente de Reutilização de software*, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro.

BRERETON P., 1999, *Evolution of Component-Based Systems*, International Workshop on Component-Based Software Engineering, Los Angeles, EUA, Maio, pp.123-126.

BURROWS C., GEORGE G. and DART S., 1996, "*Configuration Management*", Ovum Limited (Ed.), London, UK, ISBN: 1898972761, http://www.ovum.com.

CRNKOVIC I., LARSSON M., 2000, *A Case Study: Demands on Component-based Development*, Proceedings 22nd International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, June, pp: 23-31.

GAMMA, E., HELM, R., JOHNSON, R., et al., 1994, Design Patterns: Elements of Reusable Object-Oriented Software, Massachusetts, Addison Wesley.

DANTAS, C.R., "Construção de Rastros através de Sistemas de Gerência de Configuração de Software", Proposta de Tese de Mestrado, COPPE/UFRJ.

ISO 10007, "Quality Management – Guidelines for Configuration Management", disponível em http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16546, acessado em 16/06/2003.

LARSSON M., 2000, "*Applying Configuration Management Techniques to Component-Based Systems*", IT Licentiate thesis, 2000-007, MRTC Report 00/24, Uppsala University, Department of Information Technology, Dissertation for the degree of Licentiate of Philosophy in Computer Systems, Sweden, ISSN 1404-5117, December.

LEBSACK C.S., MROCZEK A.J., MUELLER C.J., 2001, Controlling Configuration Items in Component Based Software Development, Tenth International Workshop on Software Configuration Management (SCM-10) in 23rd International Conference on Software Engineering (ICSE 2001), Toronto, Canada.

LEON A., 2000, "*A Guide to Software Configuration Management*", Artech House (Ed.), ISBN: 1580530729, Cap.8.

LU ZHANG, HONG M., HONG Z., 2001, *A Configuration Management System Supporting Component-Based Software Development* 25th Annual International Computer Software and Applications Conference (COMPSAC'01) October 08-12, Chicago, Illinois, pp: 25-31.

MURTA, L.G.P., WERNER, C.M.L., 2003, "Proposta de Arquitetura de Gerencia de Configuração para Ambientes de Reutilização", Relatório Técnico, COPPE/UFRJ (em elaboração).

OLIVEIRA, H.L.R., 2003, "Evolução de Artefatos Reutilizáveis", Proposta de Tese de Mestrado, COPPE/UFRJ.

PRESSMAN, R., 2000, "Software Engineering: A Practioners Approach", ed. 5, McGraw Hill

SILVA, L.F.S., SPINOLA, R.O., WERNER, C.M.L., 2003, "Adequação das Normas ISO/IEC 9126-1 e 14598-1 para Avaliação e Certificação de COTS", Segundo Simpósio Brasileiro de Qualidade de Software, Fortaleza, Ceará (submetido para avaliação).