Gerência de Configuração no Desenvolvimento Baseado em Componentes

Leonardo Gresta Paulino Murta e Cláudia Maria Lima Werner

PESC/COPPE – Universidade Federal do Rio de Janeiro
Caixa Postal 68.511 – 21.945-970 – Rio de Janeiro – RJ – Brasil

{murta,werner}@cos.ufrj.br

Abstract. *This work presents an integrated Software Configuration Management (SCM) infrastructure for the Component-based Development (CBD). This infrastructure is composed by an SCM process tailored to CBD; a customizable change control system tightly integrated with a flexible version control system for fine-grained UML model elements; a mechanism for establishing and evolving traceability links among high level architectural elements and source code; a mechanism for deploying components on demand; and a mechanism that applies data-mining over the integrated SCM repository to discover change traces among versioned UML model elements.*

Resumo. *Este trabalho apresenta uma infra-estrutura integrada de Gerência de Configuração de Software (GCS) para o Desenvolvimento Baseado em Componentes (DBC). Essa infra-estrutura é composta por um processo de GCS adaptado ao DBC; um sistema de controle de modificações customizável, fortemente integrado a um sistema de controle de versões flexível para elementos de modelo UML em granularidade fina; um mecanismo para estabelecer e evoluir ligações de rastreabilidade entre elementos arquiteturais de alto nível e código fonte; um mecanismo para implantar componentes por demanda; e um mecanismo que aplica mineração de dados sobre o repositório integrado de GCS para detectar rastros de modificação entre os elementos de modelo UML versionados.*

1. Introdução

O paradigma de Desenvolvimento Baseado em Componentes (DBC) contorna algumas das deficiências detectadas na orientação a objetos no que tange à reutilização de software (Page-Jones, 1999). A unidade de encapsulamento utilizada (i.e., componente) é mais grossa, acarretando em um menor acoplamento com os demais componentes e maior coesão dentro do próprio componente. Além disso, os acoplamentos existentes são tratados de forma especial, devido ao uso de interfaces e conectores, visando maximizar a capacidade de substituição das partes constituintes de sistemas. Mais ainda, os componentes, interfaces e conectores podem ser vistos como importantes ferramentas para o tratamento da complexidade crescente do desenvolvimento de software. Além desses aspectos estruturais, o DBC também envolve aspectos metodológicos (Jacobson *et al.*, 1997), que colocam a reutilização em destaque no processo de desenvolvimento.

No DBC, equipes produtoras constroem e mantêm componentes reutilizáveis, enquanto equipes consumidoras adaptam e reutilizam esses componentes para construir

aplicações específicas. Devido à necessidade intransponível de manter o software após a sua liberação, a reutilização deve ocorrer de forma controlada, possibilitando que a evolução dos componentes reutilizáveis não dificulte o trabalho já complexo de desenvolvimento de componentes e de desenvolvimento de aplicações à partir de componentes. Esse controle pode ser obtido pelo uso de técnicas de Gerência de Configuração de Software (GCS), que visam apoiar a evolução de sistemas de software.

A GCS é reconhecida como um elemento crítico do processo de manutenção de software (IEEE, 1998). Apesar de existir um forte apelo para o uso da GCS durante a manutenção, a sua aplicação não se restringe somente a essa etapa do ciclo de vida do software (IEEE, 1987; Leon, 2000; Chrissis *et al.*, 2003; SOFTEX, 2006). Durante o desenvolvimento, os sistemas de GCS são fundamentais para prover controle sobre os artefatos produzidos e modificados por diferentes engenheiros de software. Além disso, esses sistemas possibilitam um acompanhamento minucioso do andamento das tarefas de desenvolvimento.

Diversas normas e modelos de melhoria de processo consideram a GCS como um processo chave para o aumento de qualidade (ISO, 1995; Chrissis et al., 2003; SOFTEX, 2006). Contudo, de acordo com o relatório sobre Qualidade e Produtividade no Setor de Software Brasileiro (MCT, 2002), referente a 2001, somente 10,4% das empresas entrevistadas (45 empresas de um espaço amostral de 431) praticavam gestão de mudanças. Mais ainda, somente 20,1% das empresas entrevistadas (85 empresas de um total de 423) faziam uso de ferramentas de GCS.

Essa situação é agravada pela falta de apoio das técnicas de GCS existentes quando aplicadas no cenário de DBC. A evolução de artefatos reutilizáveis introduz complexidade extra ao problema tratado pela GCS convencional. Essa complexidade ocorre em virtude da necessidade freqüente de adaptar o artefato antes da sua efetiva reutilização. Contudo, as várias instâncias adaptadas de um artefato são profundamente semelhantes, e, possivelmente, os mesmos defeitos ou necessidades de melhorias acontecerão em todas essas instâncias.

Desta forma, o objetivo deste trabalho consiste em analisar os problemas existentes no DBC que podem ser apoiados por técnicas de GCS e elaborar uma solução, envolvendo processos e ferramental de apoio, para minimizar o efeito desses problemas.

Este trabalho está organizado em outras 5 seções, além dessa introdução. A seção 2 detalha os problemas relacionados à GCS que surgem com a adoção do DBC. A seção 3 apresenta a abordagem proposta, seguida de uma avaliação, apresentada na seção 4. A seção 5 compara a abordagem proposta com trabalhos relacionados. Finalmente, a seção 6 conclui o trabalho apresentando contribuições, limitações e trabalhos futuros.

2. Novos Desafios da GCS no DBC

A GCS a ser aplicada ao DBC difere em vários aspectos da GCS convencional. Desta forma, o problema tratado neste trabalho pode ser subdividido em cinco partes mais específicas, que são: (1) os processos que guiam a GCS num cenário de DBC; (2) o controle de solicitações de modificação sobre os artefatos, levando em consideração a

cadeia de responsabilidades existente no DBC; (3) o controle de versões de artefatos reutilizáveis no nível de abstração do próprio artefato; (4) o gerenciamento de construção, que determina como artefatos se relacionam para estruturar sistemas; e (5) a integração dos espaços de trabalho de GCS e DBC.

Uma das características que diferem o processo de GCS convencional do processo de GCS no DBC é a responsabilidade de implementação das solicitações de modificação. No DBC, essa responsabilidade não é tão óbvia quanto no desenvolvimento convencional. Devido a possíveis cadeias de produção de componentes, compostos por outros componentes, essa responsabilidade fica distribuída entre as diversas equipes existentes no ciclo produtivo do componente. Todavia, os sistemas de controle de modificações convencionais não estão aptos para tratar esse problema. Além disso, devido à própria imaturidade das poucas propostas existentes para o processo de GCS no DBC, esse processo pode precisar ser modificado com freqüência, forçando alterações no próprio sistema de controle de solicitações.

No que tange ao controle de versões, os sistemas convencionais usualmente atuam sobre arquivos do sistema operacional. Entretanto, a maioria das normas de GCS (IEEE, 1987) recomenda uma identificação seletiva dos Itens de Configuração (ICs), que depende das características individuais dos projetos de desenvolvimento de software. Em algumas circunstâncias não é desejado, nem possível, mapear elementos de alto nível (análise e projeto) em arquivos individuais, dificultando ou inviabilizando um controle de versões adequado utilizando os sistemas convencionais sobre esses ICs. Por exemplo, seria desejável controlar a evolução das partes de um modelo UML, (e.g.: pacotes, classes, métodos, atributos e casos de uso). Contudo, com o uso dos sistemas convencionais, o modelo como um todo é versionado, pois este é armazenado em um único arquivo.

Mesmo quando é possível mapear elementos arquiteturais de alto nível para a sua implementação em arquivos do sistema operacional, a manutenção não automatizada desse mapeamento é custosa e suscetível a erros (Settimi *et al.*, 2004). Modificações tanto nos próprios elementos arquiteturais quanto na sua implementação podem levar a inconsistências. Os sistemas convencionais de gerenciamento de construção não se preocupam com a evolução do mapeamento entre a arquitetura e a sua implementação, tornando inviáveis diversos usos possíveis desse mapeamento.

Finalmente, um dos grandes desafios de manutenção de software em geral é identificar o impacto de modificações. Essa preocupação ganha maiores dimensões no DBC, pois a introdução de defeitos durante modificações em componentes pode afetar a todos os sistemas que reutilizam os componentes modificados.

3. Abordagem Proposta

A abordagem proposta, denominada Odyssey-SCM (SCM: *Software Configuration Management*), visa fornecer processos e ferramental de apoio para a aplicação de técnicas de GCS no contexto do DBC, minimizando os problemas levantados na seção 2. A abordagem Odyssey-SCM é composta por cinco subabordagens com enfoque nos subproblemas discutidos na seção 2: (1) Odyssey-SCMP (SCMP: *Software Configuration Management Process*), adaptações do processo de GCS para o contexto de DBC; (2) Odyssey-CCS (CCS: *Change Control System*), sistema de controle de

modificações com ênfase nos requisitos específicos do DBC; (3) Odyssey-VCS (VCS: *Version Control System*), sistema de controle de versões focado em artefatos de alto nível de abstração; (4) Odyssey-BRD (BRD: *Build, Release and Deploy*), gerenciamento de construção, liberação, empacotamento e implantação de componentes; e (5) Odyssey-WI (WI: *Workspace Integration*), integração dos espaços de trabalho de GCS e DBC. A Figura 1 apresenta o fluxo de trabalho proposto para a abordagem Odyssey-SCM, integrando as demais subabordagens.

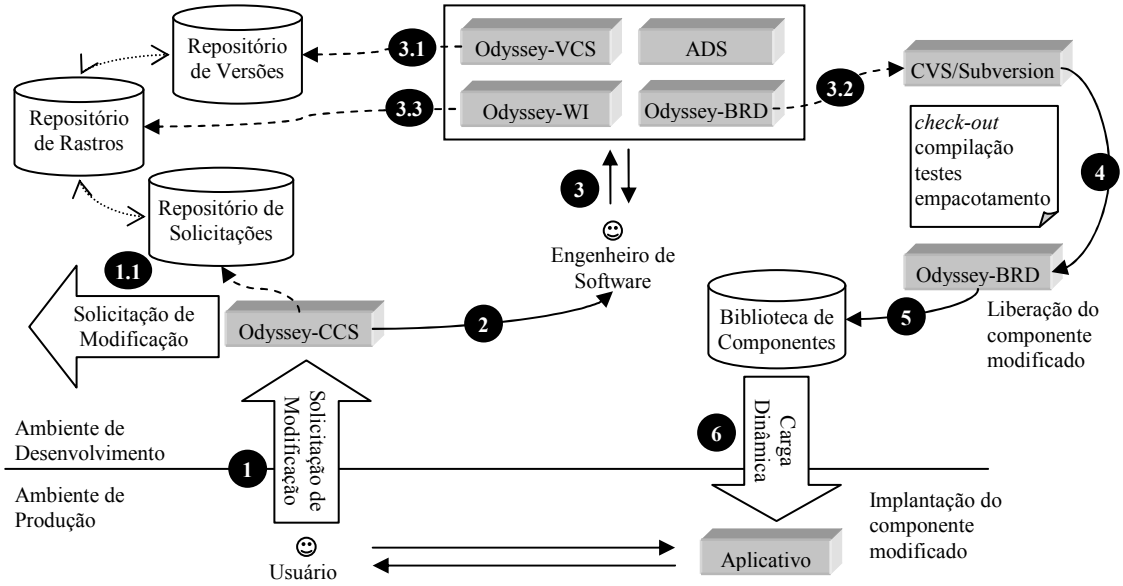

Figura 1. Fluxo de trabalho do Odyssey-SCM.

Esse fluxo de trabalho apresentado na Figura 1 segue o conjunto de passos detalhados a seguir:

1. O usuário solicita uma modificação sobre um dado aplicativo utilizando o Odyssey-CCS, de acordo com o processo Odyssey-SCMP;
   1.1. O Odyssey-CCS apóia na análise da responsabilidade de manutenção dos componentes. Caso a solicitação não seja de responsabilidade da equipe em questão, a solicitação ou parte dela é encaminhada para a equipe produtora do componente afetado;
2. Caso a solicitação ou parte dela seja de responsabilidade da equipe em questão, um engenheiro de software é designado para implementá-la;
3. Esse engenheiro de software faz uso de um Ambiente de Desenvolvimento de Software (ADS) para implementar as modificações;
   3.1. O Odyssey-VCS pode ser utilizado para acessar a versão pertinente dos modelos referentes aos componentes a serem modificados;
   3.2. Um módulo do Odyssey-BRD, denominado ArchTrace, pode ser usado para identificar e acessar o código fonte que implementa os componentes que devem ser modificados;
   3.3. Em paralelo, o Odyssey-WI pode ser utilizado para detectar outros pontos do software que devem ser modificados (análise de impacto);

4. Periodicamente (ou via comando do engenheiro de software), o Odyssey-BRD acessa o repositório de código fonte, obtendo todos os artefatos que implementam os componentes existentes. Cada componente é compilado, testado e empacotado;
5. Caso o componente esteja em um estado estável, é feita a liberação do mesmo e disponibilização em uma biblioteca de componentes;
6. Finalmente, o mecanismo de carga dinâmica do aplicativo permite que a nova versão do componente, que contempla a modificação requerida, seja instalada.

O Odyssey-SCMP consiste em uma customização do processo de GCS para o DBC, embasada nas normas ISO 10007, IEEE Std 282 e IEEE Std 1042, e nos modelos MPS.BR e CMMI. Ele tem por objetivo a colaboração entre os participantes do processo multi-nível de produção, consumo e utilização de componentes e atua na propagação de partes de solicitações de modificação em função da responsabilidade contratual de manutenção. Ou seja, como um componente é usualmente composto por outros componentes, pode ser necessário decompor solicitações de manutenção e delegar partes dessas solicitações para as equipes responsáveis. O processo Odyssey-SCMP, detalhado em anexo, foi implantado no sistema Odyssey-CCS, como mostrado na Figura 2.

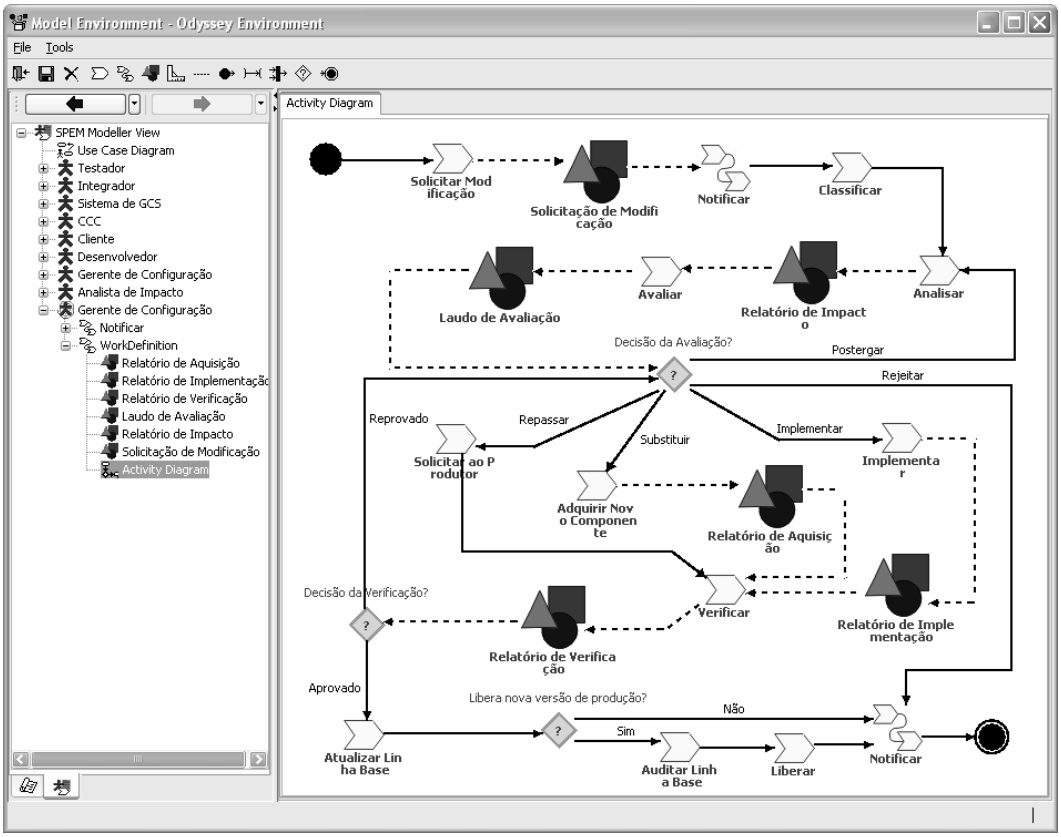

Figura 2: Processo Odyssey-SCMP modelado no Odyssey-CCS.

O Odyssey-CCS consiste em uma abordagem configurável para o controle de modificações que permite a modelagem do processo de controle de modificações (Figura 2) e que faz uso de padrões de documentação para a coleta de informações durante as atividades do processo. Essa infra-estrutura pode ser utilizada pelos participantes de ambientes de DBC de duas formas: o usuário final pode solicitar

modificações à equipe de desenvolvimento da aplicação, e a equipe de desenvolvimento da aplicação pode repassar parte das solicitações para as equipes produtoras de componentes utilizados na aplicação, dependendo de como os componentes afetados foram adaptados.

O Odyssey-VCS consiste em um mecanismo de controle de versões que atua sobre modelos descritos de acordo com o meta-modelo da UML, utilizando políticas configuráveis para as unidades de comparação e de versionamento. O conceito de unidade de comparação estabelece os artefatos na hierarquia de composição que devem ser utilizados na comparação de modificações. De forma análoga, o conceito de unidade de versionamento estabelece os artefatos na hierarquia de composição que necessitam conter informações sobre versão. Diferentemente das abordagens convencionais, que têm unidades de comparação e versionamento fixas (em arquivos e linhas, respectivamente), o Odyssey-VCS permite que esses elementos sejam configuráveis em função do próprio meta-modelo da UML. Para que isto seja possível, é necessário fazer uso da hierarquia de composição estabelecida no meta-modelo da UML. Com essa informação, juntamente com a definição das unidades de comparação e versionamento, o Odyssey-VCS é capaz de identificar quais elementos devem ser analisados no momento da detecção de conflitos e quais elementos devem ter informações de versionamento persistidas. A Figura 3 exibe os elementos de modelo UML no repositório do Odyssey-VCS (a) e depois do *check-out*, no espaço de trabalho do ambiente Odyssey (b). O ambiente Odyssey (Werner *et al.*, 2003), que consiste em um ADS orientado a reutilização, pode ser utilizados em conjunto com a abordagem proposta por ser compatível com a especificação XMI.

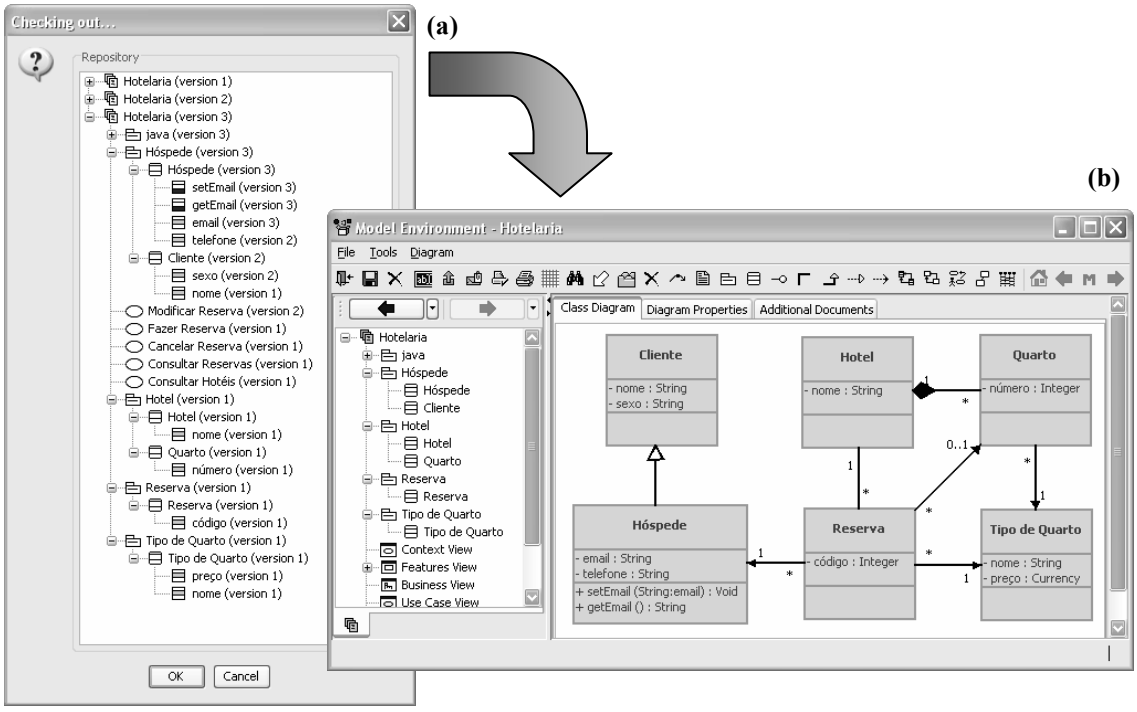

Figura 3: *Check-out* de elementos de modelo UML do Odyssey-VCS (a) para o ambiente Odyssey (b).

O Odyssey-BRD é decomposto em duas principais vertentes, que são o ArchTrace e a Carga Dinâmica. O ArchTrace consiste em um mecanismo para a

evolução das ligações de rastreabilidade existentes entre a arquitetura do software e a sua implementação através de uma infra-estrutura baseada em políticas. Esse mecanismo monitora tanto a arquitetura quanto o código fonte e detecta quando ocorrem modificações. Como resposta para as modificações, novas ligações de rastreabilidade podem ser criadas ou ligações de rastreabilidade existentes podem ser removidas. Desta forma, é possível consultar o ArchTrace em busca dos elementos de código fonte que implementam algum elemento arquitetural ou vice versa, como apresentado na Figura 4.

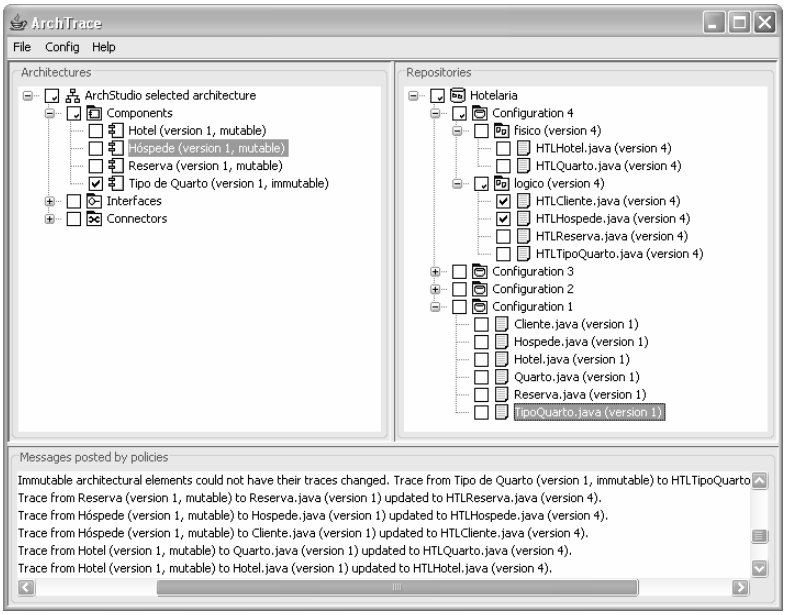

Figura 4: ArchTrace evoluindo e exibindo ligações de rastreabilidade.

Por outro lado, a Carga Dinâmica consiste em um mecanismo para o tratamento da variabilidade de aplicações em produção, possibilitando a seleção de quais componentes devem ser implantados na aplicação alvo em tempo de execução. Para que os componentes selecionados funcionem corretamente na aplicação alvo, o mecanismo calcula as dependências necessárias, recursivamente. A partir desse cálculo, o mecanismo baixa os componentes e os instala na aplicação alvo sem que sejam necessárias intervenções do usuário nem o reinício da aplicação, como exibido na Figura 5.

O Odyssey-WI faz uso de técnicas de mineração de dados sobre o repositório do Odyssey-SCM (i.e., versões de elementos de modelo UML e solicitações de modificação) para detectar ligações de rastreabilidade entre elementos UML. Essas ligações de rastreabilidade detectadas a partir de análises das modificações anteriores, devidamente contextualizadas com informações obtidas automaticamente no próprio repositório (i.e.: quem, quando, como, onde, o quê e por quê), são denominadas rastros de modificação. Rastros de modificação podem ser utilizados para sugerir modificações futuras, prevenir erros oriundos de modificações incompletas, detectar efeitos colaterais de modificações, apoiar na análise de impacto de modificações e detectar falhas de projeto devido ao alto acoplamento entre artefatos não correlatos. A Figura 6 mostra os resultados de uma consulta sobre rastros de modificação referentes à classe “Reserva”, apresentada na Figura 3.b. Esses resultados englobam indícios estatísticos sobre o relacionamento entre a classe “Reserva” e demais elementos de modelo (e.g., 57% das

modificações existentes na base afetaram tanto a classe “Reserva” quanto o caso de uso “Fazer Reserva” e 67% das modificações que afetaram a classe “Reserva” também afetaram o caso de uso “Fazer Reserva”). Além disso, os resultados também englobam indícios qualitativos sobre o relacionamento entre a classe “Reserva” e demais elementos de modelo (e.g., Lucas é responsável por uma das cinco modificações que afetaram tanto a classe “Reserva” quanto a classe “Hotel”).

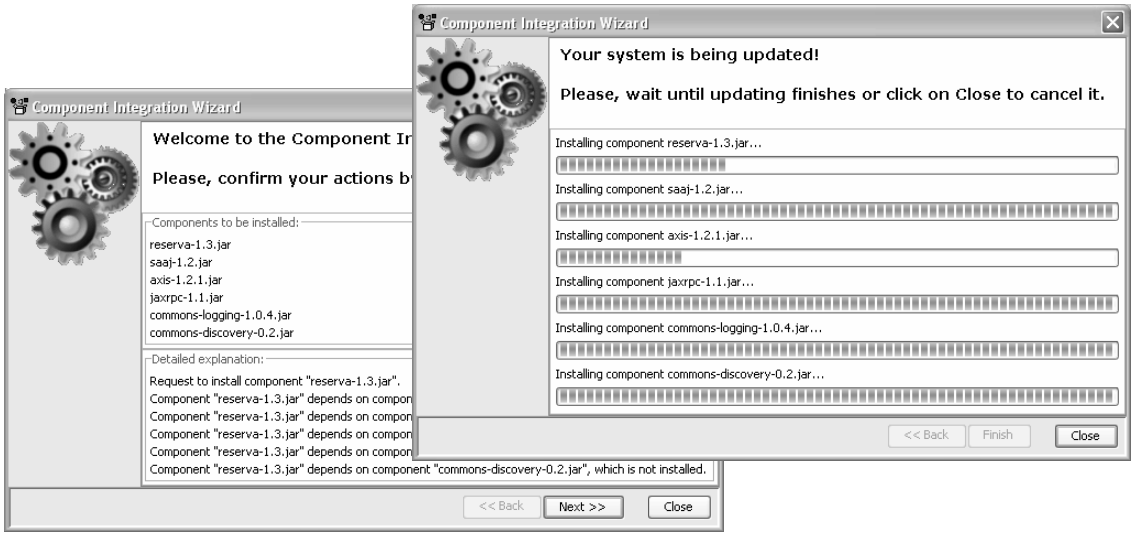

Figura 5: Carga Dinâmica de components.

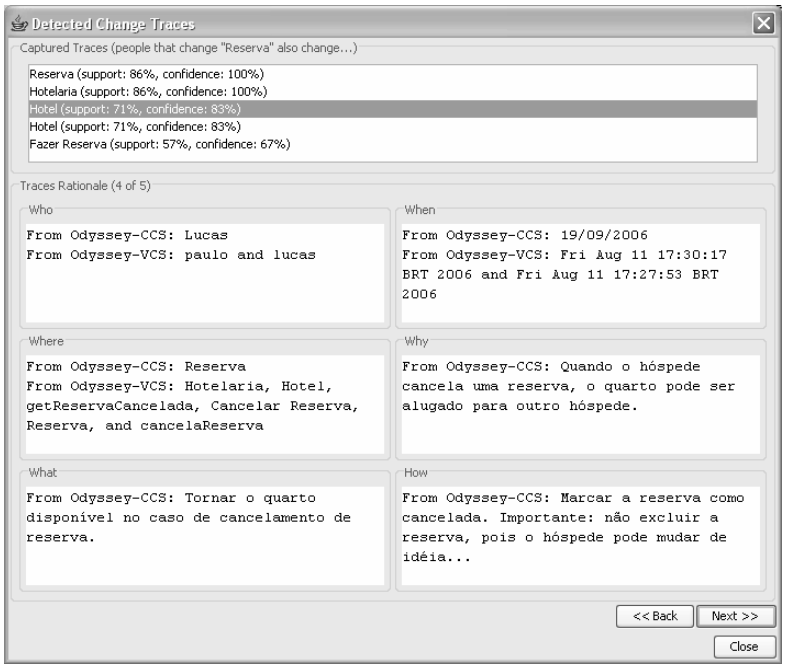

Figura 6: Odyssey-WI apresentando os rastros de modificação da classe “Reserva”.

4. Avaliação da abordagem

A abordagem Odyssey-SCM foi avaliada com foco nas subabordagens Odyssey-VCS, Odyssey-BRD e Odyssey-WI. Esta avaliação teve ênfase em aspectos de desempenho e

escalabilidade, e foram executadas através de estudos retrospectivos (i.e., estudos que analisam dados históricos para simular o comportamento do objeto de estudo naquelas situações). Os resultados iniciais são promissores, contudo, avaliações complementares sob outras dimensões do problema se tornam fundamentais caso esse protótipo acadêmico venha a ser transformado em produto.

No caso do Odyssey-VCS, foi identificada a necessidade de continuidade em pesquisas buscando formas eficientes de representação e transformação de dados semi-estruturados em estruturas orientadas a objetos. O maior gargalo identificado foi o processamento de arquivos XMI contendo modelos UML grandes. Tanto a leitura quanto a reconstrução desses arquivos demandam uma carga de processamento que inviabiliza a utilização prática da ferramenta.

Em relação ao Odyssey-BRD, as políticas atuais do ArchTrace se mostraram capazes de evoluir as ligações de rastreabilidade do ambiente Odyssey por um longo período de tempo. Entretanto, essas políticas não foram suficientes quando houve a remoção equivocada e posterior restabelecimento de diretórios, situação comum em projetos menos estáveis. Desta forma, novas políticas precisam ser projetadas para tratar a inserção de elementos novos.

O Odyssey-BRD também é composto pela carga dinâmica, que está em operação no ambiente Odyssey desde 20 de outubro de 2003. Neste período, 7 liberações candidatas foram feitas até que em 9 de maio de 2005 foi feita a sua primeira liberação estável. Desde então, outras 5 liberações estáveis foram feitas. A liberação mais recente totaliza mais de 15 componentes, carregados dinamicamente no ambiente Odyssey, sendo que esses componentes dependem de mais de 40 bibliotecas diferentes, também carregadas por demanda.

Quanto ao Odyssey-WI, os resultados obtidos foram superiores aos existentes na literatura, que tratam de código fonte (Ying *et al.*, 2004; Zimmermann *et al.*, 2004). Uma explicação possível é o fato de modelos UML serem menos suscetíveis a pequenas modificações, se comparados com código fonte, somente considerando modificações que realmente afetam a estrutura de alto nível do sistema.

5. Trabalhos Relacionados

Apesar da existência de uma ampla infra-estrutura de GCS para o desenvolvimento convencional, a integração da GCS com paradigmas específicos do desenvolvimento de software ainda é muito carente (Estublier *et al.*, 2005). O DBC é um exemplo de paradigma específico do desenvolvimento de software que necessita um maior apoio na evolução controlada de seus artefatos.

O processo MwR (Kwon *et al.*, 1999), implementado na ferramenta TERRA, apesar de possibilitar o registro e consulta sobre solicitações de modificação e componentes, não leva em consideração as diferentes versões de um dado componente, tampouco leva em consideração as equipes consumidoras que reutilizaram esse componente. Como discutido anteriormente, não é viável a detecção da responsabilidade de manutenção sem esse tipo de informação. Esses problemas foram contornados pelo Odyssey-CCS com o uso do mapa de reutilização juntamente com a configuração flexível do processo e dos formulários de coleta de informações. O KobrA (Atkinson *et*

al., 2001) é um outro trabalho importante para a manutenção de sistemas baseados em componentes. Apesar das características positivas do KobrA, relacionadas com a composição de componentes e dependências entre modificações, ele não trata a detecção da responsabilidade de manutenção. Além disso, não foi possível encontrar implementações computacionais da abordagem.

A maioria dos sistemas de controle de versão comerciais e livres atua sobre arquivos. Esses sistemas têm diversas limitações para manipular artefatos descritos via modelos de dados complexos. Algumas ferramentas CASE, como, por exemplo, Enterprise Architect e Poseidon, fazem uso desses sistemas para controlar a evolução de modelos UML, estando sujeitas aos problemas descritos na seção 2. O Odyssey-VCS pode ser visto como um sistema complementar, que possibilita o controle de versão sobre modelos UML em granularidade fina, enquanto os sistemas tradicionais tratam os demais artefatos manipulados no projeto.

Dentre as abordagens que lidam com rastreabilidade entre diferentes representações do software (Shirabad *et al.*, 2001; Antoniol *et al.*, 2002; Briand *et al.*, 2003; Marcus e Maletic, 2003; Ying *et al.*, 2004; Zimmermann *et al.*, 2004), a maioria está focada somente na detecção das ligações de rastreabilidade, em detrimento da evolução das mesmas. Desta forma, essas abordagens reconstroem as ligações de rastreabilidade de tempos em tempos, ignorando informações que poderiam apoiar numa reconstrução progressiva e contínua, como é feito pelo ArchTrace.

Além disso, alguns trabalhos fazem uso da dimensão histórica de repositórios de GCS para entender o software. Shirabad et al. (2001) utilizaram aprendizado indutivo para detectar níveis de relevância entre arquivos. Eick et al. (2001) analisaram o histórico de modificações e aplicaram índices de decaimento do código para identificar fatores de risco em sistemas. Finalmente, Zimmermann et al. (2004) evidenciaram que a mineração sobre repositórios de GCS pode ser útil para apoiar modificações futuras, detectando dependências ocultas e prevenindo modificações incompletas. Contudo, essas abordagens atuam sobre repositórios de GCS baseados em arquivos. Por essa razão, nenhuma delas apóia a detecção de rastros de modificação em elementos UML de granularidade fina. Além disso, nenhuma delas fornece indicadores da razão de surgimento dos rastros de modificação, extraídos automaticamente de uma infra-estrutura integrada de GCS. Esses dois pontos são tratados pelo Odyssey-WI.

6. Conclusão

Este trabalho apresentou uma infra-estrutura que faz uso de processos e técnicas de GCS projetadas especialmente para o cenário de DBC. Publicações e premiações obtidas durante a elaboração desse trabalho são detalhadas em http://www.cos.ufrj.br/~murta.

As principais contribuições deste trabalho são: (1) estudo das áreas de DBC e GCS, visando identificar as limitações existentes no apoio de GCS dado ao DBC; (2) definição de uma arquitetura integrada para GCS, visando prover apoio a cenários de DBC; (3) definição de um processo de GCS voltado para questões específicas do DBC; (4) especificação e implementação de um sistema de controle de modificações flexível, que possibilita a configuração do processo e das informações a serem coletadas durante a execução do processo; (5) especificação e implementação de um mapa de reutilização, que possibilita a coleta e consulta sobre informações contratuais de reutilização de

componentes; (6) especificação e implementação de um sistema de controle de versões que atua em granularidade fina sobre modelos UML, possibilitando a configuração das unidades de versionamento e comparação para cada elemento de modelo; (7) especificação e implementação de mecanismos para definição de ligações de rastreabilidade entre elementos arquiteturais e código fonte, juntamente com nove políticas que automatizam a evolução dessas ligações de rastreabilidade em resposta a modificações na arquitetura ou na sua implementação; (8) especificação e implementação de mecanismos para a propagação de comandos de GCS dos níveis arquiteturais para o código fonte, por meio do uso das ligações de rastreabilidade previamente estabelecidas; (9) especificação e implementação de uma infra-estrutura para carga de componentes por demanda no ambiente de produção, sem que seja necessário reiniciar o ambiente de produção; (10) especificação e implementação de um mecanismo para detecção de rastros de modificação entre elementos de modelo UML, via aplicação de mineração de dados; (11) contextualização dos rastros de modificação minerados por meio da coleta de informações oriundas da integração dos sistemas de controle de versões e de controle de modificações; e (12) avaliação do desempenho dos sistemas de controle de versões, de evolução de ligações de rastreabilidade e de detecção de rastros de modificação.

Durante a execução deste trabalho, algumas limitações puderam ser observadas. Dentre elas estão a necessidade de evolução do meta-modelo da UML utilizado (i.e., versão 1.4), tornando a infra-estrutura compatível com a versão 2, e a substituição do repositório MOF adotado (i.e., MDR), visando um aumento de desempenho da infra-estrutura. Além disso, alguns trabalhos futuros também puderam ser vislumbrados. Dentre eles estão a execução automática de teste nos componentes, visando assegurar a qualidade do produto continuamente, e a visualização de grandes quantidades de informação, usualmente existente em repositórios integrados de GCS.

7. Referências Bibliográficas

Antoniol, G., Canfora, G., Casazza, G.*, et al.*, 2002, "Recovering Traceability Links between Code and Documentation", *IEEE Transactions on Software Engineering (TSE)*, v. 28, n. 10 (October), pp. 970-983.

Atkinson, C., Bayer, J., Bunse, C.*, et al.*, 2001, *Component-Based Product Line Engineering with UML*, Addison-Wesley.

Briand, L.C., Labiche, Y., O'Sullivan, L., 2003, "Impact Analysis and Change Management of UML Models". In: *International Conference on Software Maintenance (ICSM)*, pp. 256-265, Amsterdam, Netherlands, September.

Chrissis, M.B., Konrad, M., Shrum, S., 2003, *CMMI: Guidelines for Process Integration and Product Improvement* Boston, MA, Addison-Wesley.

Eick, S.G., Graves, T.L., Karr, A.F.*, et al.*, 2001, "Does code decay? Assessing the evidence from change management data", *IEEE Transactions on Software Engineering (TSE)*, v. 27, n. 1 (January), pp. 1-12.

Estublier, J., Leblang, D., van der Hoek, A.*, et al.*, 2005, "Impact of Software Engineering Research on the Practice of Software Configuration Management", *ACM*

Transactions on Software Engineering and Methodology (TOSEM), v. 14, n. 4 (October), pp. 1-48.

IEEE, 1987, *Std 1042 - IEEE Guide to Software Configuration Management*, Institute of Electrical and Electronics Engineers.

IEEE, 1998, *Std 1219 - IEEE Standard for Software Maintenance*, Institute of Electrical and Electronics Engineers.

ISO, 1995, *ISO/IEC 12207 - Information technology - Software life cycle processes*, International Organization for Standardization.

Jacobson, I., Griss, M., Jonsson, P., 1997, *Software Reuse: Architecture, Process and Organization for Business Success*, Addison-Wesley Professional.

Kwon, O., Shin, G., Boldyreff, C.*, et al.*, 1999, "Maintenance with Reuse: An Integrated Approach Based on Software Configuration Management". In: *Asia Pacific Software Engineering Conference*, pp. 507-515, Takamatsu, Japan, December.

Leon, A., 2000, *A Guide to Software Configuration Management* Norwood, MA, Artech House Publishers.

Marcus, A., Maletic, J.I., 2003, "Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing". In: *International Conference on Software Engineering (ICSE)*, pp. 125-135, Portland, OR, USA, May.

MCT, 2002, *Qualidade e Produtividade no Setor de Software Brasileiro*, Ministério de Ciência e Tecnologia, Secretaria de Política de Informática, Brasília, DF, Brasil.

Page-Jones, M., 1999, *Fundamentals of Object-Oriented Design in UML*, Addison-Wesley.

Settimi, R., Cleland-Huang, J., Khadra, O.B.*, et al.*, 2004, "Supporting Software Evolution through Dynamically Retrieving Traces to UML Artifacts". In: *International Workshop on Principles of Software Evolution (IWPSE)*, pp. 49-54, Kyoto, Japan, September.

Shirabad, J.S., Lethbridge, T., Matwin, S., 2001, "Supporting Software Maintenance by Mining Software Update Records". In: *International Conference on Software Maintenance (ICSM)*, pp. 22-31, Florence, Italy, November.

SOFTEX, 2006, *MPS.BR - Melhoria de Processo do Software Brasileiro - Guia Geral (Versão 1.1)*, Associação para Promoção da Excelência do Software Brasileiro.

Werner, C.M.L., Mangan, M.A.S., Murta, L.G.P.*, et al.*, 2003, "OdysseyShare: an Environment for Collaborative Component-Based Development". In: *IEEE Conference on Information Reuse and Integration (IRI)*, pp. 61-68, Las Vegas, USA, October.

Ying, A.T.T., Murphy, G.C., Ng, R.*, et al.*, 2004, "Predicting Source Code Changes by Mining Change History", *IEEE Transactions on Software Engineering (TSE)*, v. 30, n. 9 (September), pp. 574-586.

Zimmermann, T., Weisgerber, P., Diehl, S.*, et al.*, 2004, "Mining version histories to guide software changes". In: *International Conference on Software Engineering (ICSE)*, pp. 563-572, Edinburgh, Scotland, May.

Anexo: Processo Odyssey-SCMP

Nome da Atividade:	**Preparação do ambiente de GCS.**
Descrição:	Preparar o plano de GCS, identificando quais artefatos devem ser considerados como ICs, e estabelecer a arquitetura que descreve as dependências entre esses artefatos. Neste momento as ferramentas devem ser configuradas para tratar os ICs identificados.
Pré-atividade:	-
Critérios de Entrada:	Demanda de componentes por consumidores.
Critérios de Saída:	Ambiente de GCS preparado.
Responsáveis:	Gerente de configuração.
Participantes:	Equipes consumidoras.
Produtos Requeridos:	Lista de requisitos dos componentes solicitados.
Produtos Gerados:	Arquitetura de componentes e descritor de comportamento (unidades de versionamento e comparação).
Ferramentas:	Odyssey-CCS, Odyssey-VCS, Odyssey-WI e Odyssey-BRD.
Pós-atividade:	Estabelecimento de linha base do componente.

Nome da Atividade:	**Estabelecimento de linha base do componente.**
Descrição:	Estabelecer, em marcos específicos do projeto (usualmente ao término das fases de análise, projeto e codificação), linhas bases do componente que serão utilizadas como referência para as fases posteriores.
Pré-atividade:	Preparação do ambiente de GCS.
Critérios de Entrada:	Término de fase do projeto.
Critérios de Saída:	Linha base do componente na fase estabelecida.
Responsáveis:	Gerente de configuração.
Participantes:	-
Produtos Requeridos:	ICs (produzidos na fase).
Produtos Gerados:	Linha base do componente (funcional, alocada ou de produto).
Ferramentas:	Odyssey-VCS e Odyssey-BRD.
Pós-atividade:	Solicitação de modificação ou Liberação do componente.

Nome da Atividade:	**Solicitação de modificação.**
Descrição:	Solicitar uma modificação, descrevendo o nome e a organização do solicitante, a data da solicitação, a versão dos ICs afetados, um indicativo de urgência, a necessidade e a justificativa da modificação.
Pré-atividade:	Estabelecimento de linha base do componente ou Liberação do componente.
Critérios de Entrada:	Necessidade de modificação sobre um componente.
Critérios de Saída:	Modificação solicitada.
Responsáveis:	Equipes consumidoras.
Participantes:	Gerente de configuração.
Produtos Requeridos:	Mapa de reutilização.
Produtos Gerados:	Solicitação de modificação.
Ferramentas:	Odyssey-CCS.
Pós-atividade:	Classificação da modificação.

Nome da Atividade:	**Classificação da modificação.**
Descrição:	Classificar a prioridade da modificação em relação às demais modificações solicitadas. Neste momento, é importante considerar a importância dessa modificação para as demais equipes consumidoras que reutilizam os componentes afetados.
Pré-atividade:	Solicitação de modificação.
Critérios de Entrada:	Ter uma modificação solicitada, mas ainda não priorizada.
Critérios de Saída:	Ter classificado a modificação em relação às demais modificações solicitadas.
Responsáveis:	Gerente de configuração.
Participantes:	Equipes consumidoras.
Produtos Requeridos:	Solicitação de modificação e mapa de reutilização.

Produtos Gerados:	Solicitação de modificação (classificada).
Ferramentas:	Odyssey-CCS.
Pós-atividade:	Análise de impacto da modificação.

Nome da Atividade:	**Análise de impacto da modificação.**
Descrição:	Analisar o impacto da modificação em questão em relação aos ICs afetados, alternativas, custo e cronograma de implementação, e analisar o interesse das demais equipes consumidoras pela modificação. Caso a modificação atinja componentes reutilizados, é necessário interagir com o processo de controle de modificações da equipe produtora para obter estimativas de custo e cronograma.
Pré-atividade:	Classificação da modificação.
Critérios de Entrada:	Modificação classificada como próxima a ser analisada.
Critérios de Saída:	Impacto da modificação analisado.
Responsáveis:	Analista de impacto.
Participantes:	Gerente de configuração, equipes consumidoras e equipes produtoras.
Produtos Requeridos:	Solicitação de modificação e mapa de reutilização.
Produtos Gerados:	Relatório de impacto.
Ferramentas:	Odyssey-CCS, Odyssey-WI e Odyssey-BRD.
Pós-atividade:	Avaliação da modificação.

Nome da Atividade:	**Avaliação da modificação.**
Descrição:	Avaliar a viabilidade de implementação da modificação em questão. O resultado dessa avaliação pode ser: (1) rejeitar a modificação, (2) postergar a modificação, devido à necessidade de uma análise de impacto mais profunda, (3) delegar a modificação para as equipes produtoras dos componentes afetados, (4) implementar a modificação, substituindo os componentes afetados por outros componentes equivalentes, ou (5) implementar a modificação, alterando os componentes afetados.
Pré-atividade:	Análise de impacto da modificação.
Critérios de Entrada:	Modificação analisada.
Critérios de Saída:	Modificação avaliada.
Responsáveis:	Comitê de controle da configuração.
Participantes:	-
Produtos Requeridos:	Solicitação de modificação e relatório de impacto.
Produtos Gerados:	Laudo de avaliação.
Ferramentas:	Odyssey-CCS.
Pós-atividade:	Solicitação de modificação (na equipe produtora), análise de impacto da modificação ou implementação da modificação.

Nome da Atividade:	**Implementação da modificação.**
Descrição:	Implementar a modificação em todos os níveis de abstração pertinentes (análise, projeto, código, teste, etc.). Caso a implementação seja via substituição dos componentes afetados, seguir o processo de aquisição. Caso a implementação seja via alteração dos componentes afetados, seguir o processo de solução técnica.
Pré-atividade:	Avaliação da modificação.
Critérios de Entrada:	Modificação aprovada para implementação.
Critérios de Saída:	Modificação implementada.
Responsáveis:	Engenheiros de software (analistas, projetistas, programadores, etc.).
Participantes:	Gerente de configuração.
Produtos Requeridos:	Solicitação de modificação, relatório de impacto, laudo de avaliação e ICs (afetados).
Produtos Gerados:	Relatório de implementação ou relatório de aquisição e ICs (criados/modificados).
Ferramentas:	Odyssey-CCS, Odyssey-VCS, Odyssey-WI e Odyssey-BRD.
Pós-atividade:	Verificação da modificação.

Nome da Atividade:	**Verificação da modificação.**
Descrição:	Verificar se a implementação está correta via técnicas de revisão ou testes.
Pré-atividade:	Implementação da modificação.
Critérios de Entrada:	Modificação implementada.
Critérios de Saída:	Modificação verificada.
Responsáveis:	Revisores ou equipe de testes.
Participantes:	Gerente de configuração.
Produtos Requeridos:	ICs (criados/modificados), casos e dados de teste (se pertinente).
Produtos Gerados:	Relatório de verificação.
Ferramentas:	Odyssey-CCS, Odyssey-VCS e Odyssey-BRD.
Pós-atividade:	Implementação da modificação ou atualização da linha base do componente com a modificação.

Nome da Atividade:	**Atualização da linha base do componente com a modificação.**
Descrição:	Incorporar a modificação na linha base do componente.
Pré-atividade:	Verificação da modificação.
Critérios de Entrada:	Modificação verificada.
Critérios de Saída:	Linha base do componente atualizada.
Responsáveis:	Integrador.
Participantes:	Gerente de configuração.
Produtos Requeridos:	Linha base do componente e ICs (criados/modificados).
Produtos Gerados:	Linha base do componente (atualizada).
Ferramentas:	Odyssey-CCS, Odyssey-VCS e Odyssey-BRD.
Pós-atividade:	Auditoria sobre a linha base do componente.

Nome da Atividade:	**Auditoria sobre a linha base do componente.**
Descrição:	Realizar auditoria física e funcional sobre a linha base do componente, verificando tanto a estrutura dos ICs quando a sua corretude em relação aos requisitos. Além disso, o próprio sistema de GCS também pode ser alvo de auditoria. A auditoria deve ser realizada sempre que o comitê de controle da configuração decidir gerar uma liberação a partir da linha base de um dado componente. Caso a auditoria seja executada com sucesso, a liberação pode ser gerada. Caso contrário, as não conformidades devem ser reportadas na forma de solicitações de modificação.
Pré-atividade:	Atualização da linha base do componente com a modificação.
Critérios de Entrada:	Necessidade de liberação de uma nova versão do componente.
Critérios de Saída:	Linha base do componente (auditada).
Responsáveis:	Auditor.
Participantes:	Gerente de configuração.
Produtos Requeridos:	Linha base do componente.
Produtos Gerados:	Laudo de auditoria.
Ferramentas:	Odyssey-CCS, Odyssey-VCS e Odyssey-BRD.
Pós-atividade:	Solicitação de modificação ou liberação do componente.

Nome da Atividade:	**Liberação do componente.**
Descrição:	Construir, liberar e implantar o componente.
Pré-atividade:	Auditoria sobre a linha base do componente.
Critérios de Entrada:	Linha base do componente aprovada pela auditoria.
Critérios de Saída:	Nova versão do componente implantada.
Responsáveis:	Equipe de liberação.
Participantes:	Gerente de configuração e cliente.
Produtos Requeridos:	Linha base do componente.
Produtos Gerados:	Liberação do componente.
Ferramentas:	Odyssey-BRD.
Pós-atividade:	Solicitação de modificação.