Odyssey-WI: Uma Ferramenta para Mineração de Rastros de Modificação em Modelos UML Versionados

Cristine Dantas, Leonardo Murta, Cláudia Werner

COPPE/UFRJ – Programa de Engenharia de Sistemas e Computação
Caixa Postal 68511 – CEP 21945-970 – Rio de Janeiro – RJ – Brasil

{cristine, murta, werner}@cos.ufrj.br

Abstract. *This paper presents a tool that applies data mining techniques over a software configuration management repository in order to detect change traces among UML model elements. These traces can help to: suggest and predict likely further changes, prevent incomplete changes, and support the execution of the impact analysis activity through the identification of the potential side-effects of a software change. These traces are detected together with context information that assists the analysis of its utility in the maintenance process.*

Resumo. *Este artigo descreve uma ferramenta que utiliza técnicas de mineração de dados em repositórios de sistemas da gerência de configuração de software a fim de detectar rastros de modificação entre elementos de modelo UML. Durante a manutenção e construção do software, tais rastros podem: sugerir modificações futuras, minimizar alterações incompletas e apoiar a análise de impacto, através da identificação das possíveis conseqüências de uma modificação no software. Os rastros de modificação são apresentados com uma semântica própria que auxilia a análise da sua utilidade no processo de manutenção.*

1. Introdução

A atividade de desenvolvimento e manutenção requer o entendimento do software, que representa 50% a 90% do esforço de manutenção [14]. Várias abordagens [2, 13, 16, 17] têm sido propostas com o intuito de ajudar nesse processo de entendimento, atuando em código-fonte. Mas, devido a suas atuações restritas, estas propostas não atendem a contento todas as atividades relacionadas ao desenvolvimento e manutenção do software, uma vez que o processo impõe a modificação de artefatos produzidos em diferentes fases do ciclo de vida, não somente no código fonte. Por exemplo, quando o engenheiro de software modifica um modelo UML, ele se questiona: "Quais outros artefatos do modelo devem ser alterados em virtude dessa modificação?". Encontrar relações entre artefatos do software, sem nenhum mecanismo automatizado, é uma tarefa difícil, tanto durante o desenvolvimento quanto na manutenção [16].

No entanto, a análise das informações referentes à Gerência de Configuração de Software (GCS) tem o intuito de dar subsídio à melhoria do processo que está por trás das atividades de Engenharia de Software, provendo conhecimento indireto sobre este processo. Sendo assim, os repositórios dos sistemas da GCS podem ser utilizados na descoberta de informações qualitativas e quantitativas sobre vários aspectos do

desenvolvimento de software, principalmente porque esses sistemas permitem o acompanhamento detalhado do andamento das modificações do projeto [2]. Neste caso, o uso de técnicas de mineração de dados se mostra promissor, pois permite ir além das informações explícitas existentes nos repositórios. Através da análise das modificações e versões existentes no contexto dos sistemas da GCS, torna-se possível detectar automaticamente informações implícitas nos repositórios, como relações de modificação entre artefatos de software [2, 13, 16, 17]. Essas relações, que surgiram a partir da evolução do software, podem ser chamadas, neste contexto, de rastros de modificação. No caso da UML, temos, por exemplo, relações entre classes e casos de uso.

O objetivo deste trabalho é apresentar uma ferramenta, denominada Odyssey-WI, que detecta rastros de modificação entre artefatos UML versionados, juntamente com algum significado semântico. Diferentemente de algumas abordagens [6, 12] que propõem um meta-modelo para ser instanciado manualmente com a rastreabilidade entre elementos, o Odyssey-WI tem por objetivo atuar de forma automática, descobrindo relações existentes no repositório através da mineração de dados. Maiores detalhes sobre a abordagem, de mesmo nome, que baseia a ferramenta Odyssey-WI, são encontrados em [3].

Este trabalho está organizado em 3 seções, além desta introdução, que já discutiu brevemente os trabalhos relacionados. Na Seção 2, é descrita a utilização da ferramenta proposta. Na Seção 3, a ferramenta é detalhada tecnicamente, através de discussões sobre os elementos que a compõem. Finalmente, na Seção 4, são apresentadas as contribuições deste trabalho.

2. Utilização da ferramenta

O uso de sistemas de controle de versões juntamente com ferramentas CASE permite que os artefatos de software produzidos sejam controlados de forma disciplinada. Com o objetivo de detectar rastros de modificação, a ferramenta Odyssey-WI utiliza o repositório do sistema de controle de versões Odyssey-VCS [9] e do sistema de controle de modificações Odyssey-CCS [7].

Odyssey-VCS é um sistema de controle de versões independente de ferramenta CASE, que atua sobre modelos UML, fazendo uso de uma política configurável para a unidade de versionamento. A unidade de versionamento representa os elementos que serão versionados e, portanto, irão se tornar itens de configuração quando enviados para o repositório. Neste cenário, a unidade de versionamento pode ser casos de uso, classes, atributos, operações, etc. O versionamento proposto por este sistema permite o acompanhamento da evolução individual de cada elemento do modelo.

Odyssey-CCS é um sistema configurável e flexível para o processo de controle de modificações. Ele permite que o gerente de configuração modele o processo e defina atividades, no contexto do processo, juntamente com os formulários de cada atividade. Cada formulário deve ser estruturado de forma a conter um conjunto de campos, que podem ser opcionais ou obrigatórios. Este sistema é utilizado, principalmente, na etapa inicial de preparação do ambiente e na execução do processo de controle de modificações.

Na etapa inicial de preparação do ambiente, o gerente de configuração deve configurar os sistemas Odyssey-CCS, Odyssey-VCS e a ferramenta Odyssey-WI. No

caso da ferramenta Odyssey-WI deve-se: (1) definir a precisão da ferramenta e (2) correlacionar as informações presentes no sistema Odyssey-CCS, provendo contexto para o entendimento dos rastros de modificação obtidos.

A precisão definida pelo gerente de configuração (vide Figura 1.a) pode retornar um conjunto maior ou menor de rastros de modificação. Ela pode ser baixa, trazendo relações que não são, necessariamente, freqüentes no repositório, mas que, geralmente, existem em maior número, ou alta, trazendo relações que constam no repositório com freqüência, e que, geralmente, existem em menor número. Um maior detalhamento é apresentado no final desta seção.

A semântica ou significado atribuído ao rastro de modificação existe em função do raciocínio empregado pelo engenheiro de software durante o desenvolvimento. Para aumentar o entendimento do engenheiro de software durante a manutenção, torna-se necessário coletar e organizar informações dos sistemas da GCS, de forma a obter um indício de como os rastros de modificação surgiram. Esta informação fornece subsídio para a compreensão do próprio rastro de modificação. A estrutura definida como 5W+1H [5] caracteriza o que é necessário saber para que determinada atividade ou informação seja compreendida. As questões da estrutura são: o que (*What*), quem (*Who*), quando (*When*), onde (*Where*), como (*How*) e por quê (*Why*). O gerente de configuração deve atribuir um campo do formulário do processo modelado no sistema Odyssey-CCS a cada uma das questões da estrutura (vide Figura 1.b).

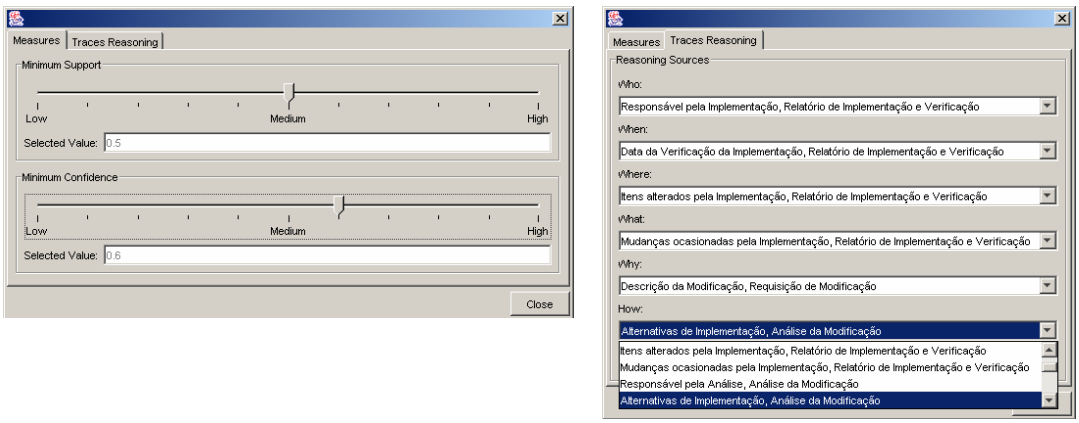

Figura 1 - (a) Configuração da precisão e (b) das questões 5W + 1H

Durante o processo de desenvolvimento, o engenheiro de software efetua *check-out* dos artefatos, realiza as modificações necessárias, e ao final, armazena os artefatos no repositório, através da operação de *check-in*. Com isso, uma nova versão dos artefatos é gerada. Esses artefatos são identificados no sistema de controle de versões como itens de configuração. A ferramenta Odyssey-WI deve ser executada sempre que o engenheiro de software necessite de orientação sobre quais artefatos deve modificar. A ferramenta, que detecta os pares de artefatos UML que foram modificados em conjunto durante o processo de desenvolvimento e manutenção, extrai do repositório as modificações realizadas no software e realiza sobre esse conjunto a mineração dos dados. A Figura 2 apresenta uma visão geral da utilização da ferramenta.

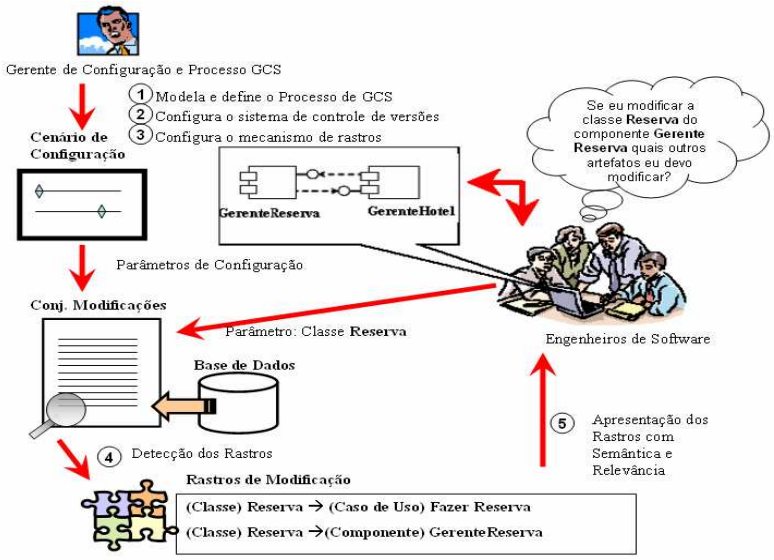

Figura 2 - Cenário de utilização da ferramenta Odyssey-WI

Ao final da mineração, a ferramenta Odyssey-WI apresenta os rastros de modificação detectados, como exemplificado na Figura 3. Na parte superior da janela são apresentados os artefatos que devem ser analisados quando a classe "Reserva" é modificada, juntamente com uma indicação de relevância para a modificação que está sendo efetuada. Na parte inferior da janela são apresentadas informações relativas às modificações passadas que embasam a existência de cada rastro de modificação.

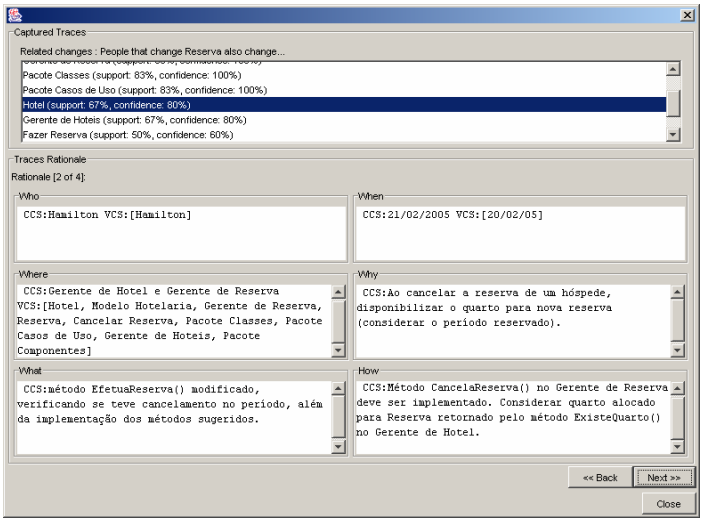

Figura 3 - Visualização dos rastros de modificação recuperados do repositório

A mineração se baseia na técnica de regras de associação e utiliza o algoritmo *Apriori* [1]. O algoritmo *Apriori* retorna todos os elementos associados ao artefato que será modificado, calculando para cada associação as medidas de suporte e confiança. O suporte *x* indica a co-ocorrência de elementos. Isto significa dizer que, do total de modificações realizadas no repositório dos sistemas da GCS, x% contém os elementos associados. A confiança *y* indica o quão forte é a presença de um elemento em função de outro, o que equivale a dizer que em y% das vezes que um elemento é modificado, outro

também é modificado. A relevância do rastro de modificação pode ser obtida através das duas medidas, que fornecem um indicador de quão freqüente é a alteração de dois elementos. A partir dessa informação e juntamente com o contexto (5W+1H), é possível avaliar a importância desse rastro na atividade que está sendo executada.

3. Detalhamento técnico

A ferramenta Odyssey-WI, juntamente com informações sobre licenciamento, está disponível em http://reuse.cos.ufrj.br. Ela é composta de um componente cliente e outro servidor. O componente servidor é independente em relação à ferramenta CASE usada para o desenvolvimento. Desta forma, o serviço remoto, referente à detecção de rastros de modificação provido pela interface do componente servidor, pode ser acessado por qualquer ferramenta CASE. Neste cenário, a camada de transporte entre os componentes cliente e servidor utiliza Web Services [15]. O formato de representação dos dados que trafegam nessa camada segue a especificação XMI [11].

O uso específico dos sistemas Odyssey-VCS e Odyssey-CCS deve-se ao fato de ambos proverem uma maior flexibilidade, seja em relação ao tipo de artefato versionado (elementos de modelo UML), seja em relação à configuração do processo de controle de modificações. Ambos os sistemas persistem as versões de modelos criados e as modificações que motivaram essas versões no repositório MDR [8]. Além de ser um repositório MOF [10], o MDR se baseia na especificação JMI (*Java Metadata Interface*) [4], que possibilita a geração de API em linguagem Java, a partir de meta-modelos MOF, e a utilização dessa API para acessar os elementos persistidos no repositório.

4. Conclusão

Esse artigo apresentou uma ferramenta que aplica técnicas de mineração de dados sobre repositórios versionados para detectar rastros de modificação entre artefatos UML. A utilização de sistemas de GCS permite que a ferramenta Odyssey-WI acesse informações já armazenadas nesses sistemas, não sendo necessário interromper o usuário, solicitando novas informações.

Com base no resultado fornecido pela ferramenta, torna-se possível: (1) a detecção de deficiências no projeto, uma vez que o engenheiro de software visualiza dependências que não foram, necessariamente, projetadas durante a concepção do software, mas que existem em função das modificações realizadas durante as atividades do processo de desenvolvimento e (2) o auxílio à atividade de análise de impacto, pois a ferramenta informa quais artefatos são usualmente afetados caso um dado artefato seja modificado.

Embora algumas abordagens [2, 13, 16, 17] apresentem contribuições similares, elas ignoram os artefatos de análise e projeto. A ferramenta Odyssey-WI se diferencia por estar no nível de modelos UML e não no nível de código-fonte, além de contribuir com informações relevantes para análise do rastro de modificação.

Referências Bibliográficas

[1] AGRAWAL, R., SRIKANT, R., 1994, *Fast Algorithms for mining association rules.* Proceedings of the 20th Very Large Data Bases Conference (VLDB), pp. 487-499, Santiago, Chile, September.

[2] BALL, T., KIM, J., PORTER, A. A., et al., 1997, *If your Version Control System Could Talk…*. Workshop on Process Modeling and Empirical Studies of Software Engineering, Boston, MA, May.

[3] DANTAS, C. R., MURTA, L. G. P., WERNER, C. M. L., 2005, *Consistent Evolution of UML Models by Automatic Detection of Change Traces*. International Workshop on Principles of Software Evolution, pp. 144-147, Lisbon, Portugal, September.

[4] DIRCKZE, R., 2002, *Java Metadata Interface (JMI) Specification - version 1.0.*, Unisys Corporation and Sun Microsystems.

[5] GUTWIN, C., GREENBERG, S., 2002, *A Descriptive Framework of Workspace Awareness for Real-Time Groupware*, Journal of Computer Supported Cooperative Work, v. 11, n. 3, pp. 411-446.

[6] LETELIER, P., 2002, *A Framework for Requirements Traceability in UML-based Projects*, International Workshop on Traceability in Emerging Forms of Software Engineering, Edinburgh, U.K, September.

[7] LOPES, L. G. B., MURTA, L. G. P., WERNER, C. M. L., 2005, *Controle de Modificações em Software no Desenvolvimento Baseado em Componentes*, Workshop de Manutenção de Software Moderna (WMSWM), Manaus, AM, Brasil, Novembro.

[8] MATULA, M., 2003, *NetBeans Metadata Repository*, http://mdr.netbeans.org.

[9] OLIVEIRA, H., MURTA, L. G. P., WERNER, C. M. L., 2004, *Odyssey-VCS: Um Sistema de Controle de Versões Para Modelos Baseados no MOF*. XVIII Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas, pp. 85-90, Brasília, DF, Brasil, Outubro.

[10] OMG, 2003, *Meta Object Facility (MOF) specification, version 1.4*, Object Management Group.

[11] OMG, 2003, *XML Metadata Interchange (XMI) Specification, version 1.1*. Object Management Group.

[12] RAMESH, B., JARKE, M., 2000, *Towards Reference Models for Requirements Traceability*, IEEE Transactions of Software Engineering, v. 27, n. 1, pp. 58-93.

[13] SAYYAD-SHIRABAD, J., 2003, *Supporting Software Maintenance by Mining Software Update Record*, Ph.D. Thesis, School of Information Technology and Engineering, University of Ottawa, May.

[14] STANDISH, T. A., 1984, *An Essay on Software Reuse*, IEEE Transactions on Software Engineering, v. 10, n. 5, pp. 494-497.

[15] W3C, 2005, *Web services Activity*, http://www.w3.org/2002/ws.

[16] YING, A. T., 2003, *Predicting Source Code Changes by Mining Revision History*, M.Sc. Dissertation, University of British Columbia, October.

[17] ZIMMERMANN, T., WEISGERBER, P., DIEHL, S., et al., 2004, *Mining version histories to guide software changes*, International Conference on Software Engineering (ICSE), pp. 23-28, Scotland, UK, May.