ORC: Uma Ferramenta de Reestruturação de Modelos baseada em Métricas para Apoiar a Reengenharia de Software Orientado a Objetos para Componentes

Ana Maria Moura[1], Aline Vasconcelos[2], Cláudia Werner[1]

[1]COPPE/UFRJ – Programa de Engenharia de Sistemas e Computação
Caixa Postal 68511 – CEP: 21945-970 – Rio de Janeiro – RJ – Brasil
[2]CEFET Campos (Centro Federal de Educação Tecnológica de Campos)
Dr. Siqueira, 273 – Pq. Dom Bosco – CEP: 28030-130 - Campos dos Goytacazes- RJ
{anammmoura, werner}@cos.ufrj.br, apires@cefetcampos.br

Abstract. *Object oriented (OO) systems can become more maintainable and reusable by restructuring their classes into cohesive groups with a well-defined functionality, generating components. This paper presents a metric based tool for the restructuring of object oriented static models that aims to obtain more cohesive and low coupled packages, i.e. component candidates, following the component based development (CBD) principles. The tool called ORC was implemented as an Odyssey plugin. Odyssey is a reuse based development environment that supports CBD, and covers development processes "for" and "with" reuse.*

Resumo. *Sistemas orientados a objetos podem se tornar mais manuteníveis e reutilizáveis através da reestruturação de suas classes em grupos coesos e com uma funcionalidade bem definida, formando componentes. Este artigo apresenta uma ferramenta que se baseia em métricas para a reestruturação de modelos estáticos orientados a objetos, visando a obtenção de pacotes mais coesos e pouco acoplados, candidatos a componentes, seguindo os princípios do desenvolvimento baseado em componentes (DBC). A ferramenta, denominada ORC, foi implementada como plugin do ambiente Odyssey, que é um ambiente de apoio à reutilização e ao DBC, englobando processos de desenvolvimento "com" e "para" reutilização.*

1. Introdução

De acordo com Cheesman e Daniels (2001), o desenvolvimento baseado em componentes (DBC) se diferencia das outras abordagens de desenvolvimento (ex: orientação a objetos) através da separação entre a especificação do componente[1] e a sua implementação, e na disponibilização dos serviços dos componentes em interfaces. As interfaces podem ser compreendidas como uma representação explícita das funcionalidades de um componente, sendo o meio de comunicação entre os mesmos. Deste modo, evita-se que mudanças na implementação de um componente afetem os demais, uma vez que os serviços continuam sendo garantidos pelas interfaces. Com isso, pode-se obter sistemas mais fáceis de manter e reutilizar. A transformação de um

[1] Componentes podem ser definidos como: "artefatos auto-contidos, claramente identificáveis, que descrevem ou realizam uma função específica e têm interfaces claras, documentação apropriada e um grau de reutilização definido" [Sametinger 1997].

software orientado a objetos (OO) para componentes visa, dessa forma, torná-lo mais manutenível e reutilizável.

Na promoção da transformação do software OO para componentes, é preciso agrupar classes de acordo com princípios de DBC. Partindo-se dos agrupamentos existentes naturalmente no software OO (i.e. pacotes), pode-se reorganizá-los de forma a torná-los mais coesos e menos acoplados, obtendo-se bons candidatos a componentes.

Neste contexto, este artigo apresenta a ferramenta ORC (*Object Restructuring to Component*) que visa apoiar engenheiros de software na reestruturação de agrupamentos de classes baseando-se em métricas de projeto OO adaptadas para o contexto de DBC [Moura *et al* 2008]. A ferramenta ORC tem como entrada um modelo de projeto detalhado atual do software. Este modelo pode ser obtido através de alguma abordagem de engenharia reversa [Bojic e Velasevic 2000; Vasconcelos 2007; Veronese e Netto 2001]. A integração com um ambiente de apoio a modelagem e ao DBC facilitará a execução da abordagem pela disponibilidade de ferramentas (ex engenharia reversa, geração de código). A ferramenta ORC está integrada ao ambiente Odyssey [Odyssey 2008], sendo possível a obtenção das vantagens mencionadas anteriormente incluindo um ambiente próprio para a manipulação dos modelos. O Odyssey é um ambiente de reutilização baseado em modelos de domínio, que oferece apoio ao DBC.

Na literatura, existem algumas ferramentas que se propõem à apoiar a reengenharia para componentes como: [Álvaro *et al* 2003] que difere deste trabalho, principalmente, por apoiar abordagens para a modernização de sistemas procedurais, e [Washizaki e Fukazawa 2005] por ser dependente da linguagem do sistema alvo e por ter foco na extração de componentes em partes do sistema.

Partindo desta Introdução, o restante do artigo está organizado da seguinte forma: a Seção 2 apresenta uma visão geral da ferramenta e sua arquitetura; a Seção 3 foca em um exemplo de utilização da ferramenta, demonstrando as suas principais funcionalidades; a Seção 4 apresenta as considerações finais e trabalhos futuros.

2. ORC - Object Restructuring to Components

A ferramenta ORC apóia a reestruturação do modelo de classes, visando que os agrupamentos de classes obtidos, candidatos a componentes, representem "bons projetos" de acordo com os princípios de DBC. Para verificar o quanto os projetos estão ou não em conformidade com alguns princípios de DBC, são coletadas métricas a partir dos modelos. Através do resultado da coleta de métricas, são sugeridas algumas reestruturações. As métricas não são coletadas todas de uma vez para que o engenheiro de software possa visualizá-las por nível de granularidade, ou seja, ele pode visualizar informações que são obtidas em diferentes níveis de detalhe (i.e. classe ou pacote). Após a geração das sugestões de reestruturação, com base nos valores das métricas, o engenheiro de software aplicará as sugestões que desejar. Este processo é iterativo, onde após um conjunto de reestruturações, segue-se um novo ciclo de coleta de métricas e geração de sugestões até que se julgue que foram obtidos bons agrupamentos baseando-se na melhoria dos valores das métricas e em seu conhecimento do domínio. A Tabela 1 apresenta, de forma resumida, as reestruturações e as métricas que as apóiam. a A lista completa pode ser encontrada em (Moura *et al* 2008).

Em síntese, os princípios do DBC adotados pela ferramenta são: não deve existir relacionamento de herança entre componentes distintos, pois um componente não deve

conhecer a implementação do outro; os componentes devem prover uma funcionalidade específica; classes generalizadas são boas candidatas a comporem componentes, pois permitem maior adaptação e extensão em novos sistemas; a comunicação inter-componentes, feita via interfaces, é muito cara em termos de tempo e recursos de plataforma e, por isso, deve-se diminuir o acoplamento entre componentes para minimizar o tráfego na rede. As métricas propostas se baseiam nesses princípios.

Tabela 1. Resumo das métricas e suas reestruturações, propostas por [Moura *et al* 2008].

Métrica: **Número de Filhos por Superclasse para cada Componente** (NOCC – *Number Of Children per Component Superclass*).	
Interpretação:	Superclasses não devem possuir subclasses em componentes distintos ao qual ela pertence, pois este relacionamento faz com que um componente conheça detalhes da implementação do outro.
Reestruturações:	• **Mover Hierarquia**
Métrica: **Número de Implementações de Interfaces por Componente** (NOIC – *Number Of Implementation per Component*).	
Interpretação:	Classes que implementam uma mesma interface estão inseridas em um contexto comum e fornecem os mesmos serviços, de modo que uni-las deixará a funcionalidade em um único componente.
Reestruturações:	• **Mover Hierarquia**
Métrica: **Acoplamento entre Classes por Componente** (CBOC – *Coupling Between Object Classes per Component*).	
Interpretação:	O acoplamento da classe com relação ao componente ao qual ela pertence deve ser igual ou maior do que o acoplamento com relação aos demais componentes.
Reestruturações:	• **Mover Classe**
Métrica: **Abstração** (A - *Abstraction*).	
Interpretação:	Um componente deve ser abstrato ou bastante abstrato, sendo mais facilmente adaptado em diferentes contextos de reutilização dentro de um domínio.
Reestruturações:	• **Extração de Superclasse** • **Extração de Interface**
Métrica: **Componentes Conectados** (ConnComp – *Connected Components*).	
Interpretação:	Grupos de classes isolados no componente podem significar que o componente possui mais de uma funcionalidade e perda de coesão.
Reestruturações:	• **Extração de Componente**
Métrica: **Acoplamento por Componente Requerido** (CRC – *Coupling per Required Component*).	
Interpretação:	Verifica quantos componentes são necessários ao utilizar cada componente, pois o número de componentes requeridos deve ser o menor possível. A comunicação inter-componentes é muito custosa.
Reestruturações:	• **Colapso de Componentes**

Em síntese, os princípios do DBC adotados pela ferramenta são: não deve existir relacionamento de herança entre componentes distintos, pois um componente não deve conhecer a implementação do outro; os componentes devem prover uma funcionalidade específica; classes generalizadas são boas candidatas a comporem componentes, pois permitem maior adaptação e extensão em novos sistemas; a comunicação inter-componentes, feita via interfaces, é muito cara em termos de tempo e recursos de plataforma e, por isso, deve-se diminuir o acoplamento entre componentes para minimizar o tráfego na rede. As métricas propostas se baseiam nesses princípios.

Espera-se que com a utilização da ferramenta ORC o engenheiro de software obtenha um modelo de pacotes e classes com um bom projeto interno para

componentes, baseado em métricas de projeto OO, adaptadas aos princípios de DBC, e sugestões de reestruturações. Como o processo é semi-automatizado, o engenheiro de software pode utilizar o seu conhecimento do domínio, junto aos valores e interpretação das métricas, para decidir quais reestruturações aplicar. O conhecimento do domínio, por parte do engenheiro de software, influencia fortemente a obtenção de agrupamentos de classes mais semânticos, visto que as métricas adotadas não avaliam semântica.

A ferramenta coleta as métricas e aplica as reestruturações selecionadas no nível de modelo. O ambiente Odyssey permite o uso de qualquer ferramenta de engenharia reversa, sendo possível importar um modelo no formato XMI.

2.1. Visão Geral da Arquitetura

A ferramenta ORC é aplicada sobre modelos de projeto do sistema alvo, tornando fundamental a sua obtenção através de alguma ferramenta de engenharia reversa, como dito anteriormente. O modelo de projeto pode ser obtido, por exemplo, pela ferramenta Ares [Veronese e Netto 2001] que é um *plugin* do ambiente Odyssey [Odyssey 2008].

A Figura 1 apresenta o funcionamento geral da ferramenta ORC. A ferramenta foi implementada na linguagem Java e, para a criação do *plugin*, a ferramenta possui uma classe para a integração com o ambiente Odyssey, que implementa uma interface padrão para o desenvolvimento de *plugins* do Odyssey, denominada "ToolAdapter". Esta classe está presente no pacote *Integration*, que é mostrado na Figura 2. O modelo recuperado através de alguma ferramenta de engenharia reversa é passado para a ferramenta ORC através do ambiente Odyssey após a importação do modelo em formato XMI, caso o mesmo não seja obtido pela ferramenta de engenharia reversa do Odyssey, i.e. a Ares. O modelo passa por uma coleta de métricas e, com base em seus resultados, são sugeridas algumas reestruturações no modelo.

Como mostra a Figura 1, a ferramenta ORC possui um módulo coletor de métricas, que coleta as métricas a partir do modelo no Odyssey, e um reestruturador de modelos, que atualiza o modelo recuperado no Odyssey, de acordo com as reestruturações selecionadas. Após a atuação do coletor de métricas, as métricas podem ser visualizadas na própria ferramenta, enquanto a cada reestruturação, o modelo reestruturado pode ser visualizado através do ambiente Odyssey.

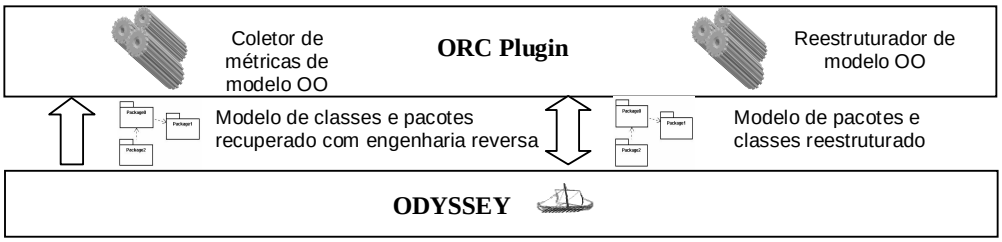


Figura 1.Visão geral do funcionamento da ferramenta ORC.

A Figura 2 apresenta uma visão geral da arquitetura da ferramenta. Ela foi implementada com um total de 6 pacotes, como mostra a Figura 2, e 20 classes. Os pacotes possuem os seguintes propósitos: classes gerais de interface gráfica que serviriam para qualquer interface gráfica, como classes para a geração de gráficos (Gui); integração com o ambiente Odyssey (*Integration*); classes específicas do reestruturador e do coletor de métricas (*Restruct*) Internamente ao pacote *Restruct* há mais 3 pacotes, a saber: classes de interface gráfica específicas para a estratégia de reestruturação, como

as telas da ferramenta apresentadas na seção 3 (Gui); classes do coletor de métricas (*MetricCollector*); e classes próprias para o reestruturador (*Restructurer*).

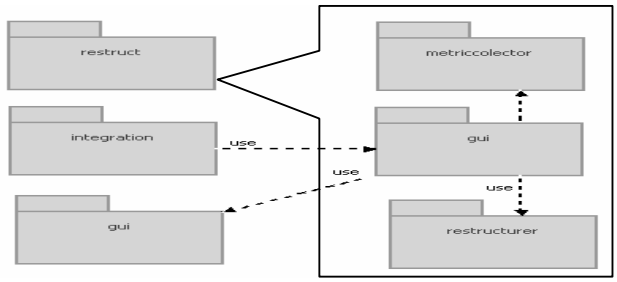


Figura 2. Visão geral dos elementos arquiteturais da ferramenta ORC.

Entre as limitações atuais da ferramenta ORC, encontra-se a utilização somente através do ambiente Odyssey. Este fator é determinante devido à importação XMI, mencionada anteriormente, que depende da compatibilidade entre o Odyssey e outras ferramentas de modelagem. Apesar desta limitação, utilizar a ferramenta ORC traz alguns benefícios como: a coleta automática das métricas, minimizando erros de coleta; a análise automática dos valores para encontrar as reestruturações necessárias; e a possibilidade de desfazer automaticamente quaisquer reestruturações aplicadas.

3. Exemplo de Utilização

A ferramenta ORC é aplicada sobre modelos, portanto para ilustrar a utilização da ferramenta, primeiramente, foi obtido o modelo OO estático do projeto do sistema alvo. O projeto possui uma série de agrupamentos de classes, que são alvos da reestruturação. O sistema alvo é a ferramenta TraceMining [Vasconcelos 2007], que é uma ferramenta utilizada na recuperação de arquitetura de referência através de técnicas de mineração de dados e está presente no ambiente Odyssey.

Inicialmente, o engenheiro de software é informado sobre o objetivo da estratégia de reestruturação que a ferramenta apóia. Além do objetivo, ele é informado sobre as métricas que serão disponibilizadas para ele. Esta explicação geral da estratégia de reestruturação é abordada em detalhes em [Moura *et al* 2008].

O segundo passo da abordagem é a seleção do nível de granularidade (i.e. classe ou pacote) que determina o conjunto de métricas e sugestões de reestruturação que serão visualizados. Deste modo, o engenheiro de software pode reestruturar o modelo em duas etapas. A ferramenta apresenta uma explicação sobre o nível de granularidade, incluindo informações sobre a ordem de utilização das métricas mais adequada, que visa à minimização do número de reestruturações a serem aplicadas, e neste ponto o engenheiro de software seleciona o nível por onde deseja começar a reestruturar. Durante todo o tempo de reestruturação, o engenheiro de software poderá retornar a este ponto e alterar o nível de granularidade sem necessitar recomeçar a reestruturação ou perder as reestruturações já aplicadas.

O próximo passo é a realização da coleta de métricas, que é feita automaticamente, e a geração de sugestões de reestruturações, que também é automatizada. O resultado da coleta de métricas é visualizado através de gráficos, como mostra a Figura 4. Cada métrica possui uma página (Aba) específica que é identificada pelo nome da métrica e sua sigla. A visualização das métricas é feita através de gráficos

que apresentam os valores das métricas coletadas em barras e os valores de referência em linha, quando não representam um intervalo de valores, caso contrário os valores de referência serão representados por faixas de valores delimitadas pelo intervalo. Na Figura 4, pode-se visualizar a apresentação dos valores como descrito anteriormente.

A tela da Figura 4 apresenta ainda um painel com as informações sobre a métrica (i.e. descrição e interpretação) e opções de reestruturar ou desfazer reestruturações aplicadas. A métrica mostrada nesta Figura é a métrica componentes conectados (ConnComp – *Connected Components*), que indica o número de grupos de classes isolados por pacote. O resultado desta métrica indica uma possível perda de coesão no pacote, como o valor 6 para o pacote GUI, e a reestruturação proposta é a divisão do pacote em outros pacotes, um para cada grupo isolado que foi encontrado.

A sugestão de reestruturações é visualizada por métrica através de uma opção própria, como dito anteriormente, podendo ser visualizada na Figura 5. A lista de sugestões tem uma visualização particular que permite a aplicação de uma ou mais sugestões de reestruturação a partir de uma mesma métrica. Como apresentado na Figura 5, as sugestões de reestruturação envolvem a extração de componentes para cada pacote que apresentou valor acima do valor de referência. Para compreender melhor as reestruturações sugeridas, são apresentadas algumas informações particulares de cada reestruturação (i.e. mecanismo de funcionamento e forte recomendação ou não de utilização para a aderência aos princípios de DBC), como descrito em [Moura *et al* 2008]. As reestruturações sugeridas na Figura 5 são fortemente recomendadas para que os componentes possuam funcionalidades específicas. Toda reestruturação aplicada pode ser desfeita em qualquer momento de reestruturação. O engenheiro de software deve escolher o ponto até o qual deseja retornar e todas as reestruturações realizadas após aquele ponto deverão ser desfeitas também.

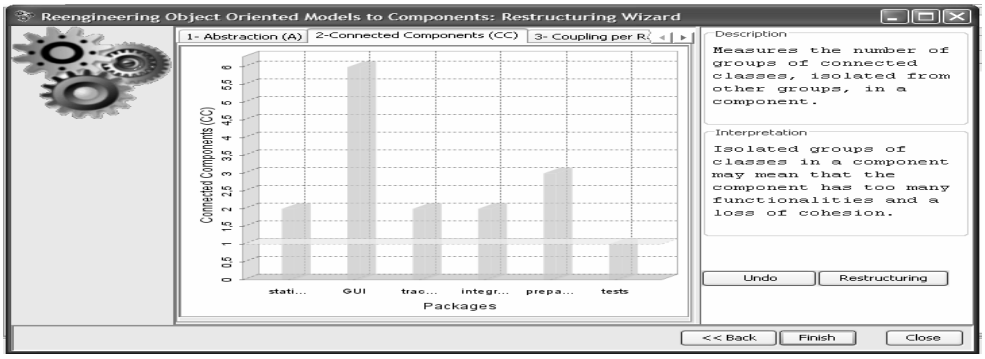


Figura 4. Tela de apresentação de resultados da coleta de métricas.

A sugestão de reestruturações é visualizada por métrica através de uma opção própria, como dito anteriormente, podendo ser visualizada na Figura 5. A lista de sugestões tem uma visualização particular que permite a aplicação de uma ou mais sugestões de reestruturação a partir de uma mesma métrica. Como apresentado na Figura 5, as sugestões de reestruturação envolvem a extração de componentes para cada pacote que apresentou valor acima do valor de referência. Para compreender melhor as reestruturações sugeridas, são apresentadas algumas informações particulares de cada reestruturação (i.e. mecanismo de funcionamento e forte recomendação ou não de utilização para a aderência aos princípios de DBC), como descrito em [Moura *et al*, 2008]. As reestruturações sugeridas na Figura 5 são fortemente recomendadas para que

os componentes possuam funcionalidades específicas. Toda reestruturação aplicada pode ser desfeita em qualquer momento de reestruturação. O engenheiro de software deve escolher o ponto até o qual deseja retornar e todas as reestruturações realizadas após aquele ponto deverão ser desfeitas também.

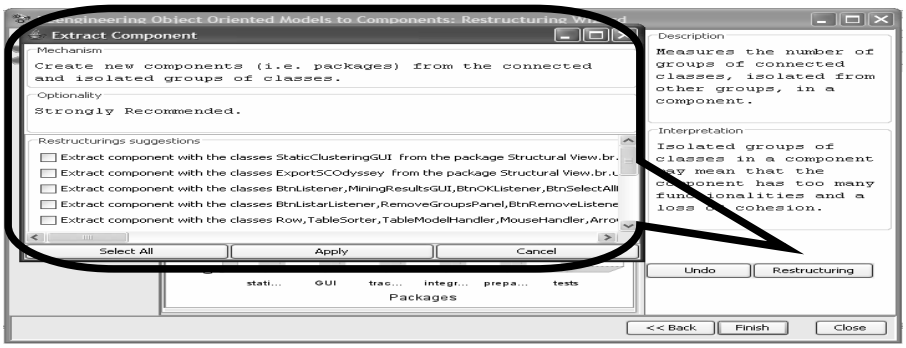
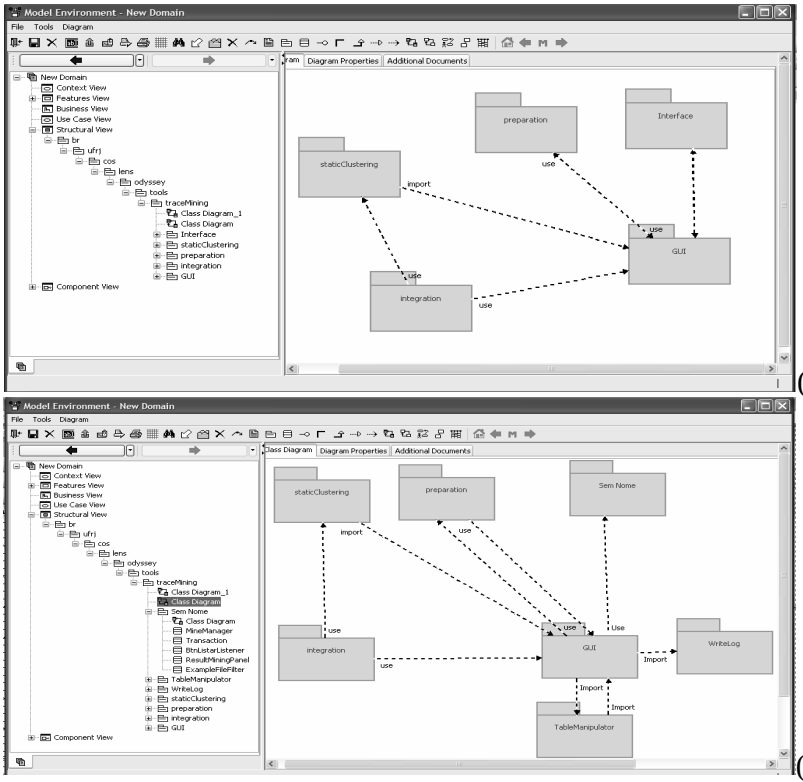


Figura 5. Tela de aplicação de sugestões de reestruturações.



(a)

(b)

Figura 6. Modelo de pacotes antes (a) e depois (b) da reestruturação.

Na Figura 6a, encontra-se o modelo do sistema alvo antes da aplicação da estratégia de reestruturação. O modelo resultante após a aplicação da estratégia pode ser visualizado no próprio ambiente Odyssey, assim como o modelo inicial. O modelo resultante é apresentado na Figura 6b e possui algumas melhorias como: a necessidade de isolamento, indicada pela métrica ConnComp, com relação a algumas funcionalidades, resultando nos pacotes *TableManipulator* e *Writelog,* que poderão ser mais facilmente reutilizadas; e pacotes mais coesos como o pacote GUI, que englobava a função de manipulação de tabela que foi extraída para o pacote *TableManipulator*.

4. Conclusões e Trabalhos Futuros

Neste artigo, foi apresentada a ferramenta ORC para apoiar uma estratégia de reestruturação de modelo de pacotes e classes, que é parte de uma abordagem de reengenharia de software OO para componentes. Como contribuições, estão a possibilidade de tornar os sistemas OO mais reutilizáveis e manuteníveis, utilizando informações baseadas em métricas OO adaptadas aos princípios de DBC para a tomada de decisão. As métricas são apresentadas de forma gráfica e possibilitam guiar a aplicação de reestruturações do modelo com apoio ferramental. Como trabalho em andamento, encontra-se a definição de um *checklist* para apoiar a verificação do modelo reestruturado, a fim de assegurar a preservação das suas funcionalidades. Como trabalhos futuros estão o desenvolvimento da estratégia para a geração de componentes e suas interfaces a partir dos modelos reestruturados e a realização de estudos de caso para avaliar a ferramenta ORC, envolvendo sistemas reais e em maior escala.

A ORC está disponível na página do projeto [Odyssey 2008]. Ela é compatível com o ambiente Odyssey, na release 1.6.0, disponível no mesmo link.

5. Referências

Alvaro, A., Lucredio, D., Garcia, V.C., Prado, A. F., (2003). Orion-RE: a component-based software reengineering environment. Em: Proceedings of the 10th Working Conference on Reverse Engineering (WCRE). Victoria, BC, Canada. pp. 248 – 257. 13-16 Novembro.

Bojic, D., Velasevic, D. (2000). A Use-Case Driven Method of Architecture Recovery for Program Understanding and Reuse Reengineering. In: 4th European Software Maintenance and Reengineering Conference. Zurique, Suiça, pp. 23-31, Fevereiro.

Cheesman, J., Daniels, J. (2001). UML components: a Simple Process for Specifying Component-based Software. Addison-Wesley Longman Publishing.

Moura, A. M. M., Vasconcelos, A. P. V., Werner, C. M. L., (2008). Uma Estratégia de Reestruturação de Modelos baseada em Métricas para Apoiar a Reengenharia de Software Orientado a Objetos para Componentes. Workshop de Manutenção Moderna de Software (WMSWM). Florianópolis, SC, Brasil, pp. 1-9, Junho.

Odyssey. (2008). In: http://reuse.cos.ufrj.br/odyssey, acessado em 17/03/2008.

Sametinger, J. (1997). Software Engineering with Reusable Components. Springer-Verlag New York, Inc.

Vasconcelos, A. (2007). Uma Abordagem de Apoio à Criação de Arquiteturas de Referência de Domínio Baseada na Análise de Sistemas Legados. Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

Veronese, G. O., Netto, F. J. (2001). ARES: Uma Ferramenta de Auxílio à Recuperação de Modelos UML de Projeto a partir de Código Java. Projeto Final de Curso de Bacharelado em Informática- Instituto de Matemática/UFRJ, Outubro.

Washizaki, H., Fukazawa, Y. (2005). A Technique for Automatic Component Extraction from Object-Oriented Programs by Refactoring, Science of Computer Programming, Vol.56, No.1-2, pp.99-116, 2005.