Um Framework para Recuperação Arquitetural Independente de Plataforma

Luis Paulo T. de Oliveira e Sandro S. Andrade

Grupo de Pesquisa em Sistemas Distribuídos, Otimização, Redes e Tempo-Real (GSORT)
Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBa)
Av. Araújo Pinho, nº 39 - Canela - Salvador - BA - CEP: 40.110-150

{luisoliveira, sandroandrade}@ifba.edu.br

Resumo. *Técnicas para recuperação arquitetural viabilizam a obtenção de conhecimento sobre os artefatos que compõem um software. Entretanto, em meio a um grande número de abordagens existentes, a prospecção de plataformas simples e flexíveis para o desenvolvimento e utilização de diferentes técnicas de recuperação se torna uma atividade importante. Este trabalho apresenta um framework para recuperação arquitetural caracterizado pela flexibilidade em relação a: i) plataforma de desenvolvimento utilizada no software cuja arquitetura será recuperada; ii) algoritmo de recuperação utilizado; e iii) notação de modelagem aplicada na representação da arquitetura recuperada. Um exemplo de instanciação do framework é brevemente descrito ao final do artigo.*

1. Introdução

Compreender as decisões e táticas utilizadas no projeto da arquitetura de um *software* é fundamental para um melhor suporte às atividades de manutenção, evolução e configuração de aplicações [Taylor et al. 2009]. Decisões arquiteturais impactam diretamente o grau de atendimento dos requisitos não-funcionais, a manutenção da integridade conceitual e a adoção de estratégias de reuso.

Diante da importância das atividades de recuperação de decisões arquiteturais, iniciativas para o desenvolvimento e sofisticação de tais tarefas têm se tornado mais frequentes, gerando um número crescente de técnicas de recuperação arquitetural. Cada iniciativa difere das demais em relação aos elementos arquiteturais considerados na recuperação, público-alvo dos modelos recuperados, artefatos utilizados como entrada do processo de recuperação, notação utilizada na descrição dos modelos, dentre outros aspectos.

Apesar do aumento do número de abordagens de recuperação arquitetural propostas, ainda existem restrições à escolha de alguma técnica particular. A maioria das soluções dependem de uma plataforma de desenvolvimento específica. Adicionalmente, a técnica de recuperação pode não apresentar o conjunto exato de informações desejáveis sobre a arquitetura recuperada. Ainda, a notação utilizada pode não ter a expressividade e formalidade requeridas. Por fim, a identificação de conectores como entidades de primeira-classe também é um fator importante na escolha por uma determinada técnica.

Sob esta perspectiva, este artigo apresenta um *application framework* para recuperação arquitetural independente de plataforma de desenvolvimento, de notação de modelagem e de algoritmo de recuperação adotados. Seus *hot-spots* podem ser facilmente configurados para atender estas diferentes necessidades, viabilizando a implementação

de novas formas de recuperação, o suporte a novas plataformas de desenvolvimento e a utilização de novas notações de modelagem. Além disso, um *hot-spot* para definição de *code snippets* para a recuperação de conectores é também disponibilizado.

O *framework* foi implementado como um componente da ferramenta DuSE-MT[1] [Andrade and Macêdo 2013] – *software* livre para modelagem e análise de arquiteturas desenvolvido em C++ e Qt. O DuSE-MT oferece um ambiente para criação e manipulação de modelos de *software*. Um núcleo independente de metamodelo é disponibilizado junto com implementações das linguagens UML (*Unified Modeling Language*) e MOF (*Meta-Object Facility*).

2. Recuperação Arquitetural

Compreender a arquitetura de um *software* e garantir a sua consistência são fatores críticos para a manutenção de sistemas. Entretanto, modificações arbitrárias – que impactam decisões arquiteturais previamente tomadas – frequentemente provocam desvios e/ou erosões arquiteturais. Em algum momento, é necessário recuperar a arquitetura descritiva de um sistema para fins de análise ou comparação com a sua arquitetura prescritiva. As decisões tomadas sobre a estrutura do *software*, bem como suas características comportamentais, são recuperadas a partir dos artefatos de implementação disponíveis. Dentre as principais técnicas de recuperação arquitetural existentes, destacam-se a clusterização hierárquica e o *graph pattern matching*.

A clusterização hierárquica [Maqbool and Babri 2007] utiliza código-fonte ou documentação para recuperar visões arquiteturais estruturais. O objetivo é formar grupos de itens/entidades semelhantes entre si. As técnicas baseadas em *graph pattern matching*, por sua vez, analisam o código-fonte da aplicação para derivar grafos que representam as dependências entre entidades. Técnicas de *matching* são então empregadas para verificar similaridades com arquiteturas de referência previamente definidas.

3. O Framework Proposto

O *framework* foi projetado de modo a apresentar flexibilidade em relação a três aspectos principais (*hot-spots*): plataforma de desenvolvimento utilizada; algoritmo de recuperação arquitetural adotado; e notação de modelagem arquitetural utilizada na representação da arquitetura.

Conforme apresentado na Figura 1, o *framework* define uma *engine* genérica para recuperação definindo quatro classes/interfaces que viabilizam a implementação dos *hot-spots* indicados. É possível instanciar o *framework* a partir de heranças/implementações de tais classes/interfaces. A classe `ArchitectureRecoveryAlgorithm` define a API a ser implementada pelos diferentes algoritmos de recuperação arquitetural. O método *recoverAlgorithm()* é um *template method* que delega a execução do método abstrato *run()* a subclasses e, em seguida, executa o método *represent()* do objeto que implementa a notação de modelagem utilizada. Um algoritmo particular de recuperação deve implementar o método *run()* e popular as estruturas de dados que representam os componentes, conectores e ligações (*attachments*) encontrados. O *framework* define dois *bridges* de `ArchitectureRecoveryAlgorithm` com o *backend* de recuperação e com a notação de modelagem utilizada.

[1] http://duse.sf.net

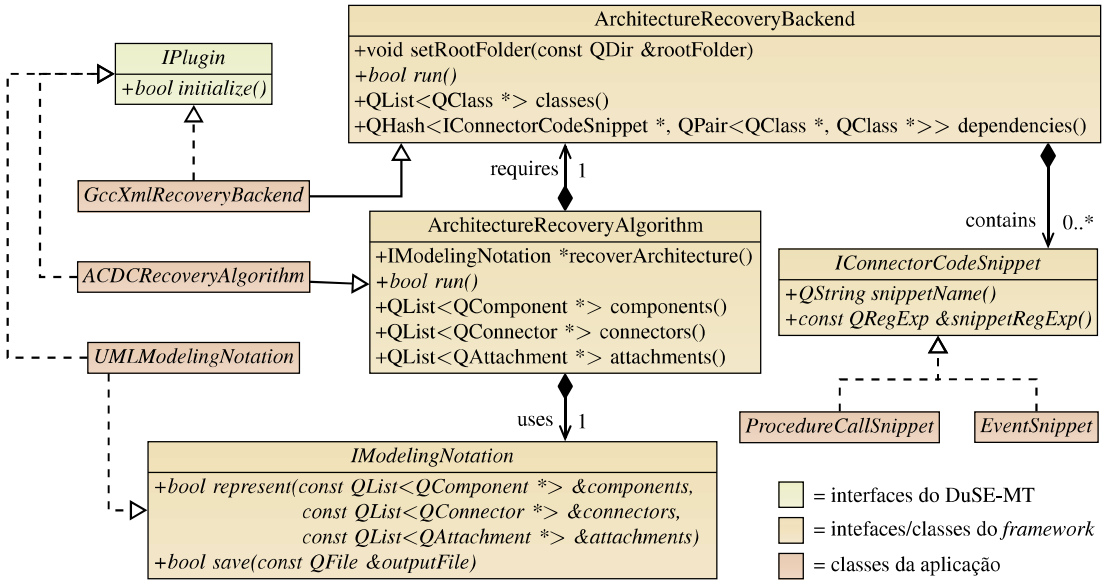

Figura 1. Interfaces e classes definidas no *framework*.

`ArchitectureRecoveryBackend` é a interface responsável por desacoplar, dos algoritmos de recuperação, os mecanismos utilizados para analisar artefatos descritos em diferentes plataformas de desenvolvimento. Linguagens de programação diferentes requerem o uso de diferentes *parsers* para obtenção das informações elementares que subsidiam a execução dos algoritmos de recuperação arquitetural. Dessa forma, a implementação da classe `ArchitectureRecoveryBackend` deve ser direcionada para a plataforma na qual o *software* foi desenvolvido, informando como o método *run()* analisa o código-fonte e obtém informações básicas sobre classes e suas dependências.

Conectores de *software* desempenham um papel crucial no projeto e análise de arquiteturas. Eles mediam a interação entre componentes e são fundamentais para a satisfação de requisitos não-funcionais. Sendo assim, `IConnectorCodeSnippet` permite a definição de expressões regulares que analisam o código-fonte em busca de trechos que descrevem a utilização de um conector específico. O objetivo é flexibilizar a detecção de conectores não previstos, viabilizando a sua representação.

`IModelingNotation` define a API a ser implementada de modo a suportar diferentes formas de representação dos artefatos arquiteturais recuperados. As notações utilizadas variam desde as mais informais, tais como linguagem natural e gráficos *ad-hoc*, até as notações mais formais e rigorosas, como as ADLs (*Architecture Description Languages*). Uma determinada notação deve implementar o método *represent()* informando como os componentes, conectores e ligações deverão ser representados na notação de modelagem em questão. Adicionalmente, o método *save()* pode ser implementado para realizar a serialização do modelo em algum formato previamente definido, como por exemplo serialização de modelos UML em XMI (*XML Metadata Interchange*).

Todas as implementações concretas de *hot-spots* foram desenvolvidas como *plugins* do DuSE-MT e, portanto, implementam a interface `IPlugin` definida pela ferramenta. Este aspecto é importante para permitir a fácil instalação de novos algoritmos, *backends* e notações, bem como a configuração *on-the-fly* das implementações a serem

utilizadas em uma determinada recuperação.

4. Instanciação do Framework

O *framework* proposto neste trabalho foi instanciado de modo a realizar a recuperação de arquiteturas de sistemas implementados em C++/Qt. Para isto, um *backend* de recuperação baseado na ferramenta GCC-XML [King 2014] foi desenvolvido, em conjunto com *code snippets* para detecção de conectores dos tipos *Procedure Call* e *Event*.

O algoritmo de recuperação ACDC [Tzerpos 2001] foi implementado, motivado pelo seu desempenho e boa precisão de recuperação [Garcia et al. 2013]. Uma implementação de notação de modelagem para geração de modelos UML foi também desenvolvida como parte da aplicação. O *framework* tem sido avaliado em relação a: *i)* adequação de seus *hot-spots* às variações de recuperação arquitetural existentes; e *ii)* facilidade de compreensão e instanciação por desenvolvedores. Resultados preliminares indicam que seis algoritmos de recuperação são facilmente suportados na solução. Experimentos controlados estão sendo atualmente realizados para verificar/rejeitar a hipótese de que o *framework* traz melhorias na produtividade e pode ser facilmente utilizado, quando comparado à implementação de recuperação arquitetural sem o uso de *frameworks* ou com *frameworks* correlatos.

5. Conclusões e Trabalhos Futuros

Este artigo apresentou o projeto e implementação de um *framework* para recuperação arquitetural independente de plataforma, algoritmo de recuperação e notação de modelagem das arquiteturas recuperadas. Um exemplo de instanciação do *framework* para recuperação de arquiteturas de sistemas implementados em C++ foi também apresentado.

Trabalhos futuros incluem a definição de mecanismos mais sofisticados para detecção de conectores *composite*, projeto de *hot-spots* de menor granularidade para facilitar a implementação de algoritmos específicos de recuperação e melhor suporte a algoritmos de clusterização através da disponibilização prévia de métricas de similaridade.

Referências

[Andrade and Macêdo 2013] Andrade, S. S. and Macêdo, R. J. d. A. (2013). A search-based approach for architectural design of feedback control concerns in self-adaptive systems. In *Proceedings of the 7th IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, SASO 2013, Philadelphia, PA, USA. IEEE.

[Garcia et al. 2013] Garcia, J., Ivkovic, I., and Medvidovic, N. (2013). A comparative analysis of software architecture recovery techniques. In *IEEE/ACM 28th Intl. Conference on Automated Software Engineering (ASE)*, pages 486–496. IEEE.

[King 2014] King, B. K. (2014). GCC-XML. http://gccxml.github.io/. Acesso: 08/04/2014.

[Maqbool and Babri 2007] Maqbool, O. and Babri, H. (2007). Hierarchical clustering for software architecture recovery. *IEEE Trans. on Software Eng.*, 33(11):759–780.

[Taylor et al. 2009] Taylor, R. N., Medvidovic, N., and Dashofy, E. M. (2009). *Software Architecture: Foundations, Theory, and Practice*. Wiley Publishing.

[Tzerpos 2001] Tzerpos, V. (2001). *Comprehension-Driven Software Clustering*. PhD thesis, University of Toronto.