PREViA: Uma Abordagem para a Representação Visual da Evolução de Modelos Arquiteturais de Software

Marcelo Schots[1], Leonardo Murta[2], Cláudia Werner[1]

[1]Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ
Caixa Postal 68.511, Rio de Janeiro, RJ, 21945-970
{schots, werner}@cos.ufrj.br

[2]Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria 156, Niterói, RJ, 24210-240
leomurta@ic.uff.br

Nível: Mestrado
Ano de ingresso: 2008
Previsão de conclusão: Julho de 2010
Aprovação da proposta de dissertação (qualificação): Maio de 2009

Abstract. *Software systems and its architectures evolve over time due to several reasons. The consistency between the developed applications (represented by emerging architectures) and its conceptual architecture must be kept anytime. The PREViA approach (Procedure for Representing the Evolution through Visualization of Architectures) aims at improving perception and comprehension of the evolution of software architecture models through a mechanism for visualizing this evolution; such visualization represents the result of the comparison between versions of these models.*

Keywords. *Software Evolution, Software Architecture, Software Visualization, Traceability*

Resumo. *Sistemas de software e suas arquiteturas evoluem no decorrer do tempo devido a diversas razões. A consistência entre as aplicações desenvolvidas (representadas por arquiteturas emergentes) e sua arquitetura conceitual deve ser mantida em qualquer instante do tempo. A abordagem PREViA (Procedimento para a Representação da Evolução por meio da Visualização de Arquiteturas) visa auxiliar a percepção e a compreensão da evolução de modelos de arquitetura de software a partir de um mecanismo de visualização desta evolução; tal visualização representa o resultado da comparação entre versões destes modelos.*

Palavras-chave. *Evolução de Software, Arquitetura de Software, Visualização de Software, Rastreabilidade*

1. Introdução

Arquiteturas de software são modificadas no decorrer do tempo à medida que os sistemas são construídos. Isto pode fazer com que a arquitetura conceitual (ou seja, a arquitetura planejada) comece a divergir da arquitetura emergente (que representa o que se encontra efetivamente implementado em código fonte). A inconsistência entre estas arquiteturas pode ocasionar problemas, especialmente no contexto de processos de manutenção e reengenharia.

A abordagem apresentada neste trabalho objetiva a criação de um mecanismo de visualização da evolução de modelos arquiteturais de software a partir da comparação de suas versões (armazenadas, por exemplo, em repositórios de gerência de configuração). Pretende-se que, com o uso desta abordagem, seja possível obter uma melhor compreensão e percepção da aderência da arquitetura emergente com relação à arquitetura conceitual e das diferenças entre versões das arquiteturas, o que pode vir a auxiliar na tomada de decisões de projeto e de implementação do software.

O presente trabalho está organizado em outras cinco seções. A Seção 2 contém alguns conceitos que são importantes no contexto deste trabalho. A abordagem proposta é apresentada na Seção 3. A Seção 4 contém alguns trabalhos relacionados. Por fim, na Seção 5, são feitas algumas considerações finais, bem como a descrição do estágio atual do trabalho e os próximos passos.

2. Contextualização

A arquitetura de um software envolve a descrição dos elementos a partir dos quais os sistemas são construídos, as interações entre estes elementos, os padrões que direcionam sua composição e as restrições sobre estes padrões [Shaw & Garlan 1996]. As diferentes representações de um sistema, cada uma sob uma diferente perspectiva, são denominadas de visões arquiteturais. Há várias abordagens para definir, descrever e identificar visões arquiteturais (e.g. [Krutchen 1995] e [Soni *et al.* 1995]). Uma das possíveis formas para documentar tais visões é através da notação UML (*Unified Modeling Language*) [OMG 2009], voltada para sistemas orientados a objetos.

Na medida em que mudanças ocorrem no ambiente e surgem demandas por novas funcionalidades ao longo do ciclo de vida do software, podem ser necessárias alterações nos requisitos do software; tais alterações, por sua vez, implicam muitas vezes em mudanças nas descrições arquiteturais que representam estes requisitos. Adicionalmente, os programadores podem vir a seguir caminhos distintos dos que foram considerados pelos projetistas de software.

Situações como estas podem fazer com que a arquitetura conceitual (ou seja, a arquitetura planejada nas fases iniciais do ciclo de vida do software) comece a divergir da arquitetura emergente (isto é, a arquitetura que representa o que se encontra efetivamente implementado em código fonte – obtida, por exemplo, através de engenharia reversa) [Knodel *et al.* 2006]. Esta divergência pode representar, por exemplo, elementos da arquitetura conceitual que ainda não foram implementados, ou mesmo elementos implementados – presentes na arquitetura emergente – que não foram descritos na arquitetura conceitual.

Visando evitar posteriores impactos negativos, especialmente com relação a processos de manutenção e reengenharia, é importante que seja preservada a consistência entre as aplicações desenvolvidas e sua arquitetura conceitual em qualquer instante do tempo [Völter 2007].

Neste cenário, os conceitos e mecanismos de gerência de configuração de software (GCS) surgem como uma forma de se obter um melhor controle da evolução do software. Porém, no contexto da evolução de arquiteturas, tal controle não é uma atividade trivial. Mesmo utilizando mecanismos de GCS, as informações geradas pela maioria dos sistemas de controle de versão não são muito intuitivas e podem demandar muito tempo e esforço para que sejam compreendidas [Lintern *et al.* 2003].

As técnicas de visualização de software podem ser utilizadas para fornecer uma melhor percepção destas informações, visando obter em menor tempo e com menor esforço o entendimento do processo de evolução arquitetural. Estas técnicas permitem apresentar grandes quantidades de informação através de representações visuais que tornam "visíveis" os artefatos e o comportamento do software [Ball & Eick 1996] e visam proporcionar uma forma mais simples e intuitiva de compreensão do significado dos dados, auxiliando a gerência da complexidade do software [Cemin 2001].

3. Abordagem proposta – PREViA

A abordagem PREViA (Procedimento para a Representação da Evolução por meio da Visualização de Arquiteturas) tem como objetivo criar um mecanismo de visualização da evolução de modelos arquiteturais de software, obtendo informações sobre esta evolução a partir da comparação de suas versões armazenadas em fontes de dados versionados, tais como repositórios de gerência de configuração. Esta comparação pode ser feita sob três diferentes perspectivas: (1) entre duas versões da arquitetura conceitual, (2) entre duas versões da arquitetura emergente, ou (3) entre uma versão da arquitetura emergente e uma versão da arquitetura conceitual.

A fim de auxiliar a compreensão e a percepção desta evolução, serão utilizados conceitos e técnicas de visualização de software. Os focos de visualização são a aderência da arquitetura emergente à arquitetura conceitual no decorrer do tempo e as diferenças encontradas entre duas versões distintas no processo de evolução. Em ambos os focos, deseja-se obter a percepção das diferenças existentes entre as arquiteturas.

Pretende-se transmitir a idéia de "sobreposição" da arquitetura emergente sobre a arquitetura conceitual no decorrer do tempo. Uma das possíveis formas é a exibição da arquitetura conceitual em um nível alto de transparência (que possa ser configurado dentro de um conjunto de escalas de cores, sujeito a restrições de visibilidade), como se fosse uma marca d'água (*watermark*) na tela. Desta forma, na medida em que houver um aumento no nível de similaridade de um elemento descrito na arquitetura conceitual e encontrado (em implementação) na arquitetura emergente, esta irá "preenchendo" a arquitetura conceitual, realçando possíveis divergências encontradas. No que diz respeito às diferenças entre versões de arquiteturas, serão realçados os elementos em que for detectada alguma diferença.

Considerando que (1) idealmente, uma arquitetura conceitual não deve conter detalhes de implementação e deve ser independente de decisões específicas de

tecnologia, e que (2) a arquitetura emergente – derivada do código fonte – possivelmente conterá elementos que são específicos da implementação, torna-se então necessário identificar quais elementos da arquitetura emergente se encaixam na última situação. A execução desta tarefa de forma automatizada é bastante suscetível a erros, o que prejudicaria a efetividade da comparação entre modelos arquiteturais. Por outro lado, a separação de elementos de implementação dos elementos conceituais em um modelo emergente pode não ser trivial e, conseqüentemente, demandar bastante tempo e esforço, o que torna inviável o processo de identificação manual.

Sendo assim, é vislumbrada uma identificação semi-automática, que consiste na automatização do processo de identificação da similaridade dos elementos, permitindo que associações entre elementos conceituais e emergentes possam ser estabelecidas ou removidas manualmente (e.g., se elementos de implementação da arquitetura emergente forem incorretamente identificados como conceituais), de forma que elementos identificados como específicos de implementação não sejam incluídos na comparação. Para evitar retrabalho, o rastro desta identificação deve ser mantido entre as versões.

Este trabalho também utiliza os conceitos de **precisão** (*precision*) e **cobertura** (*recall*), provenientes da área de recuperação de informação. No contexto da abordagem, as medidas de precisão e cobertura podem indicar, respectivamente, a **corretude** e a **completude** de um determinado instante do desenvolvimento (ou seja, de uma dada versão do software) com relação ao que foi planejado na arquitetura conceitual. Os cálculos serão efetuados da seguinte forma:

$$\text{Precisão} = \frac{n_{ci}}{n_i - n_{ii}} \quad \text{(i)} \qquad\qquad \text{Cobertura} = \frac{n_{ci}}{n_c} \quad \text{(ii)}$$

Nas fórmulas (i) e (ii), n_{ci} é o número de elementos conceituais implementados, n_i é o número total de elementos implementados, n_{ii} é o número de elementos implementados identificados como específicos de implementação (isto é, que não foram incluídos na comparação e por isso devem ser subtraídos de forma a não interferir no cálculo da precisão) e n_c é o número total de elementos conceituais. Estas medidas são aplicáveis a elementos em diferentes níveis de abstração, desde que sua hierarquia esteja bem definida/configurada. Considerando, por exemplo, uma classe UML, o cálculo considera como elementos os atributos, métodos e relacionamentos associativos e hierárquicos presentes nesta classe.

O número total de elementos implementados é o número de elementos presentes em uma determinada versão (i.e. o número de elementos conceituais implementados somado ao número de elementos implementados que não constam no modelo conceitual). Já o número total de elementos conceituais é o número de elementos que pertencem ao modelo conceitual (i.e. o número de elementos conceituais implementados somado ao número de elementos conceituais que não foram implementados).

Visando à implementação da abordagem PREViA, será estendido o EvolTrack [Cepeda *et al.* 2008], um mecanismo de extração e visualização do ciclo de evolução de projetos de software. Pretende-se reutilizar/evoluir os conectores de fontes de dados e estender um componente de visualização para representar, conforme o contexto, as diferenças entre as arquiteturas ou a sobreposição da arquitetura emergente à conceitual. Apesar de a abordagem PREViA ter sido projetada para ser genérica e extensível,

optou-se por implementar a representação de arquiteturas em diagramas de classes UML (que representam o projeto arquitetural em baixo nível de abstração), a título de demonstração da funcionalidade da ferramenta.

A abordagem PREViA consta dos seguintes módulos: *model provider* (que utilizará o conector de fonte de dados do EvolTrack), *model analyzer* (que inclui o coletor de métricas e o comparador), *model marker* (que aplicará as métricas conforme o modelo), *model tracer* (que armazenará o rastro do "tipo" de elemento selecionado) e *visualizer* (que inclui os visualizadores de diferenças e de conformidade arquitetural).

4. Trabalhos Relacionados

O trabalho desenvolvido por Ohst *et al.* (2003) visa à detecção e visualização de diferenças existentes entre versões de diagramas UML. A abordagem proposta por estes autores para a visualização das diferenças entre dois diagramas se assemelha à abordagem PREViA no que diz respeito à representação visual das diferenças em um único diagrama. Os autores, porém, não trataram especificamente da comparação entre arquiteturas conceitual e emergente. Além disso, a abordagem não prevê a exibição da seqüência temporal de todas as versões com suas diferenças capturadas.

A ferramenta SAVE (*Static Architecture Visualization and Evaluation*) [Knodel *et al.* 2006] tem como propósito a avaliação e visualização de arquiteturas estáticas, com foco em verificações de conformidade entre os relacionamentos arquiteturais a partir da comparação efetuada entre a arquitetura conceitual e a emergente (sendo esta obtida através de engenharia reversa, utilizando-se de estratégias de transformação do código para modelo). Um aspecto não tratado pela abordagem SAVE é o armazenamento dos diversos modelos obtidos em diferentes instantes do tempo, de forma a permitir a navegação pela evolução da arquitetura como uma seqüência de modelos. Para prover esta funcionalidade, a abordagem PREViA fará uso do EvolTrack, que armazena os diversos instantes da arquitetura, possibilitando a reprodução e análise destes instantes.

5. Considerações Finais

Possíveis divergências e inconsistências entre as aplicações desenvolvidas e sua arquitetura conceitual podem vir a causar impactos negativos nos processos de desenvolvimento (em particular, no contexto de manutenção e reengenharia). A abordagem PREViA, apresentada neste trabalho, visa auxiliar a percepção e a compreensão da evolução das arquiteturas de software por meio da criação de um mecanismo de visualização da evolução de modelos arquiteturais.

Espera-se como contribuição que, ao compreender o processo de evolução do sistema a partir de sua arquitetura, seja obtida uma melhor percepção de quão aderente está o desenvolvimento com relação ao projeto da arquitetura conceitual. Quando se percebe, na versão final do sistema, um nível baixo de similaridade com a arquitetura conceitual, é possível que sejam identificados problemas no processo de elaboração das arquiteturas conceituais e/ou na implementação destas arquiteturas.

No que diz respeito à implementação da abordagem, o *model analyzer* já detecta as diferenças entre modelos, enquanto o *model marker* está parcialmente implementado. Após a finalização deste, pretende-se implementar o *model tracer* e evoluir os módulos *model provider* e *model visualizer*.

Como forma de avaliar a abordagem proposta, planeja-se executar um estudo de observação. A hipótese de pesquisa deste trabalho é que o uso da abordagem PREViA permita melhor percepção da evolução de modelos arquiteturais, através da identificação das diferenças nestes modelos conforme o foco de visualização. Os participantes serão divididos em dois grupos homogêneos entre si, e deverão identificar informações relativas à evolução arquitetural de dois projetos, A e B. Um dos grupos receberá o projeto A sem a aplicação da abordagem e, em seguida, o projeto B com a aplicação da abordagem, enquanto o outro grupo receberá o projeto B sem a aplicação da abordagem e, em seguida, o projeto A com a aplicação da abordagem. Será considerado o tempo gasto para cada etapa e o número de informações identificadas.

Agradecimento

Os autores agradecem o apoio financeiro da CAPES e do CNPq para a realização deste trabalho.

Referências

Ball, T. and Eick, S. (1996). "Software Visualization in the Large", *IEEE Computer*, 29, 4, pp. 33-43, April.

Cemin, C. (2001), *Visualização de Informações Aplicada à Gerência de Software*, Tese de D.Sc., Instituto de Informática, UFRGS, Porto Alegre, Rio Grande do Sul, Brasil.

Cepeda, R. S. V., Murta, L. G. P., and Werner, C. M. L. (2008). "Visualizando a Evolução de Software no Desenvolvimento Distribuído". *INFOCOMP (UFLA)*, S. Ed., pp. 81-90.

Knodel, J., Muthig, D., Naab, M. and Lindvall, M. (2006). "Static Evaluation of Software Architectures", In: *Proceedings of the Conference on Software Maintenance and Reengineering (CSMR)*, pp. 279-294, IEEE Computer Society, Washington, DC, March.

Kruchten, P. (1995). "Architectural Blueprints - The 4+1 View Model of Architecture", *IEEE Software*, 12, 6, pp. 42-50, November.

Lintern, R., Michaud, J., Storey, M., and Wu, X. (2003), "Plugging-in Visualization: Experiences Integrating a Visualization Tool with Eclipse". In: *ACM Symposium on Software Visualization (SoftVis)*, pp. 47-ff, San Diego, California, June.

Ohst, D., Welle, M., and Kelter, U. (2003). "Differences between versions of UML diagrams". In: *ESEC/FSE'03*, 28, 5, pp. 227-236, Helsinki, Finland, September.

OMG (2009), *Unified Modeling Language (UML), Superstructure, version 2.2*, formal/2009-02-02, Object Management Group.

Shaw, M. and Garlan, D. (1996), *Software Architecture - Perspectives on an Emerging Discipline*, Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Soni, D., Nord, R. L., and Hofmeister, C. (1995), "Software architecture in industrial applications", In: *Proceedings of the 17th International Conference on Software Engineering* (ICSE'95), pp. 196-207, ACM, New York, NY, April.

Völter, M. 2007. "Software Architecture Documentation in the Real World". Tutorial. In: *22nd International Conference on Object-Oriented Programming, Systems, Languages, and Applications* (OOPSLA 2007), ACM, Montreal, Canada, October.