ICMC USP SÃO CARLOS

Uma Abordagem de Reestruturação de Sistemas Baseada em Requisitos de Qualidade Pré-Estabelecidos

RELATÓRIO CIENTÍFICO PARCIAL - 01/06/2012 a 15/03/2013
Apresentado à Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP

Número do Processo:	FAPESP 2012/05168-4
Período:	Junho/2012 a Março/2013

Bolsista:	Rafael Serapilha Durelli (rdurelli@icmc.usp.br)
Orientador:	Prof. Dr. Márcio E. Delamaro (delamaro@icmc.usp.br)

USP - São Carlos
Janeiro de 2012

Resumo

Relatório Científico Parcial apresentado à FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) com o objetivo de elucidar as atividades realizadas pelo bolsista Rafael Serapilha Durelli durante o primeiro período de vigência da bolsa concebida sob o Processo Número 2012/05168-4. O referido período teve inicio em Junho de 2012 e foi finalizado em Março de 2013. Além disso, este relatório também descreve as atividades que estão em andamentos, bem como as atividades a serem realizadas no próximo período. Vale ressaltar que a partir de resultados preliminares alguns artigos científicos foram elaborados e publicados.

1 Introdução

Este relatório tem por objetivo apresentar as atividades realizadas pelo bolsista Rafael Serapilha Durelli durante o período de Junho/2012 a Março/2013, referente à bolsa de doutorado concebida pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) sob o Processo Número 2012/05168-4.

É importante salientar que o trabalho em questão tem sido desenvolvido no Departamento de Ciências da Computação e Estatística do Instituto de Ciência Matemáticas e de Computação (ICMC) da Universidade de São Paulo (campos São Carlos/SP). Este trabalho se insere no contexto do grupo de pesquisas em Engenheira de Software, sob a orientação do Prof. Dr. Márcio Eduardo Delamaro. Além disso, este trabalho esta sendo executado em colaboração com o grupo de engenharia de software da Universidade Federal de São Carlos (UFSCAR)[1]. Mais especificamente em colaboração com o Prof. Dr. Valter Vieira de Camargo[2], o qual tem grande experiência na área de engenharia de software com ênfase no desenvolvimento de frameworks no contexto da programação orientada a aspectos e reuso de software.

Por fim, ressalta-se que o bolsista criou um vínculo científico com o *Institut National de Recherche en Informatique et en Automatique* (INRIA), onde irá realizar um ano de

[1] http://dc.ufscar.br
[2] http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=S819089

doutorado sanduíche sobre orientação do Prof. Dr. Nicolas Anquetil[3] o qual tem grande experiência na área de manutenção e reengenharia de software.

2 Plano de Trabalho

Nesta seção são apresentados os resumos tando do plano de trabalho quanto das atividades conduzidas durante esse primeiro período de vigência da bolsa. Adicionalmente, apresenta-se o cronograma inicialmente proposto.

2.1 Resumo do Plano Inicial

Muitas organizações utilizam sistemas que foram desenvolvidos há anos e que já foram submetidos a diversas atividades de manutenção para se adequar a novos requisitos, acomodar alterações tecnológicas e satisfazer novos processos de negócio. Sistemas que não foram projetados de forma adequada para acomodar constantes alterações em seus requisitos, tendem a ter suas arquiteturas corrompidas e rapidamente tornam-se obsoletos, trazendo dificuldades para o crescimento da organização em ambientes competitivos. Quando isso ocorre, esses sistemas são comumente conhecidos como "sistemas legados" **?**.

De acordo com **?** a partir do momento em que um sistema começa a ser utilizado, ele entra em um estado contínuo de mudança. Tais sistemas tendem a se tornarem obsoletos em vista das novas tecnologias que são disponibilizadas ou em consequência de manutenções que são feitas sem planejamento. Além das correções de erros, as mudanças mais comuns que os sistemas sofrem são migrações para novos paradigmas e extensões em sua funcionalidade para atender a novos requisitos dos usuários.

Os problemas mais comuns de sistemas legados são: (*i*) tipicamente são implementados com tecnologias obsoletas fazendo com que a manutenção se torne custosa e difícil, (*ii*) usualmente não possuem documentações, e quando possuem, não estão atualizadas e (*iii*) devido a falta de documentação, integrar sistemas legados à outros sistemas tende a ser um

[3]http://rmod.lille.inria.fr/web/pier/team/Nicolas-Anquetil

processo difícil, lento e propício a erros. No entanto, sistemas legados tem geralmente uma missão importante dentro de uma companhia pois representam/armazenam informações de suma importância e assim não podem simplesmente serem descartados.

Outro problema é que muitas tarefas de manutenção necessitam de grandes modificação no código-fonte, as quais são normalmente feitas manualmente e assim tendem a gerarem efeitos colaterais em outros módulos do sistema. Além disso, em geral, processos de manutenção são iniciados sem uma clara especificação do que se quer alcançar, dificultando a identificação se o problema foi resolvido completamente.

Uma das técnicas disponíveis na literatura para melhorar a qualidade de sistemas legados é submetê-los a um processo de reengenharia que é o processo no qual "transformações" são realizadas no sistema com o intuito de melhorar sua estrutura sem alterar seu comportamento original Fowler et al (1999).

Usualmente reengenharia está ligada com transformações as quais seguem catálogos de refatorações, tais como as refatorações propostas por Fowler et al (1999). Transformação é a atividade em que um conjunto de mudanças são efetivamente realizadas no código-fonte, objetivando melhorar sua estrutura e até mesmo atender a novos requisitos **?**. Para auxiliar essa atividade algumas abordagens existem na literatura Bisbal et al (1999); Tilley (1995); Fowler et al (1999).

Similarmente várias abordagens foram propostas para auxiliar a reengenharia de sistemas legados **??**. Apesar da existência de todas essas abordagens, de acordo com **?** mais da metade dos projetos que aplicam reengenharia falham ao lidar com desafios específicos. Segundo tal autor, tanto a carência de padronização durante a atividade de reengenharia quanto a falta de apoio computacional efetivo são os principais problemas que acarretam esse grande número de falhas. A falta de um processo padronizado é um problema durante a atividade de reengenharia, a qual é realizada de maneira totalmente *ad hoc*. Adicionalmente, o código-fonte não deve ser o único artefato do software a possuir padronização, uma vez que o mesmo não representa/contêm todas as informações necessárias de um sistema.

Portanto, a atividade de reengenharia deve possuir diretrizes que auxiliem a padronização de todos os artefatos do software, e.g., código-fonte, dados, regras de negócios, etc.

Uma alternativa à reengenharia tradicional e que vem sendo investigada atualmente é a Modernização Orientada à Arquitetura (*Architecture Driven Modernization* - (ADM)), a qual permite realizar análises de descoberta de conhecimento e refatorações utilizando os princípios da abordagem de Desenvolvimento Orientado a Modelos (*Model-Driven Development* - (MDD)), ao invés de efetuá-los diretamente no código-fonte **?**.

Uma forte característica da ADM é o oferecimento de um conjunto de metamodelos padronizados para representar artefatos de um sistema legado. Entre esses metamodelos tem-se o principal ativo da OMG, denominado metamodelo de Descoberta de Conhecimento (*Knowledge Discovery Meta-Model* - KDM). KDM é um metamodelo para instanciar e modelar todos os artefatos de um sistema legado e pode ser utilizado como um metamodelo independente para auxiliar o engenheiro de software durante a atividade de modernização de sistemas legados. Entre Maio de 2004 e Agosto de 2005, mais de trinta organizações colaboraram para o desenvolvimento e revisão da padronização do KDM. Em meados de 2007, a OMG oficializou a versão 1.0 do KDM, e hoje o KDM encontrasse na versão 1.3 e é reconhecido internacionalmente como uma padronização (ISO/IEC 19506) para ser utilizado durante a atividade de modernização de sistemas legados **?**.

Este projeto de doutorado tem quatro principais motivações. A primeira é a carência de catálogos de refatorações para o metamodelo KDM. Existem documentos **???** que mostram que refatorações para esse metamodelo ainda não existem. Neste contexto, é importante criar catálogos de refatorações para o KDM com objetivo de padronizar o processo de modernização e aumentar a interoperabilidade entre ferramentas que utilizam o KDM para auxiliar o processo de modernização.

A segunda motivação é a ausência de abordagens que permitam analisar diferentes cenários de modernização antes de efetivamente realizar transformações no sistema legado. A maior parte das abordagens disponíveis na literatura realizam as refatorações diretamente no código-fonte. Assim, tais abordagens inviabilizam a possibilidade de gerar vários siste-

mas modernizados somente para averiguar qual deles é o mais adequado de acordo com os modelos de referência. No contexto deste projeto, modelos de referência são especificações do que se espera que o sistema legado atenda após ser modernizado, por exemplo, uma arquitetura orientada a serviço.

A terceira motivação é a falta de meios para se especificar tais modelos de referência. Portanto, muitas tarefas de reengenharia são iniciadas sem um entendimento claro dos problemas atuais e sem uma definição precisa das melhorias esperadas após as refatorações; fazendo com que não sejam efetivas na solução dos problemas existentes. O que as abordagens de refatoração atuais fazem é aplicar o mesmo conjunto de casos de teste nas duas versões do sistema e averiguar se a funcionalidade não foi alterada Demeyer et al (2004); Demeyer (2005); Monteiro e Fernandes (2005). Entretanto, isso não garante que os problemas foram resolvidos.

Por fim, existe é a necessidade de um apoio computacional efetivo durante a atividade de reengenharia de um sistema legado. Como ressaltado anteriormente, mais da metade dos projetos que aplicaram reengenharia em um sistema legado falharam devido a ausência de um apoio computacional efetivo **?**. A falta de padronização durante a atividade de reengenharia também acarreta esse grande número de falhas. Sem uma padronização a atividade de reengenharia tente a ser realizada de maneira totalmente *ad hoc*, o que pode atrasar e gerar gastos durante a atividade em questão.

Tendo elucidado os problemas relacionados com a reengenharia de software, o objetivo deste projeto é desenvolver um ambiente que recomende cenários alternativos de modernização com base em modelos de referência. Adicionalmente, objetiva-se desenvolver catálogos de refatorações para o metamodelo KDM. Com a utilização desses catálogos será possível fazer com o que sistemas legados sejam refatorados para uma arquitetura orientada a serviços ou outras arquiteturas candidatas. Do mesmo modo será possível que interesses transversais de um sistema legado sejam refatorados para se tornarem mais modular, por exemplo, utilizando o paradigma orientada a aspectos Kiczales et al (1997).

2.2 Cronograma

3 Resumo das Atividades Realizadas no Período

Nesta seção são apresentadas e detalhadas as atividades realizadas durante o primeiro período de bolsa, correspondente aos meses de Junho/2012 a Março/2013. Em resumo, foram cursadas um conjunto de disciplinas para a obtenção dos créditos necessários do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional do ICMC. Além disso, estudou-se a abordagem Modernização Dirigida à Arquitetura (em inglês - *Architecture-Driven Modernization* - ADM), bem como seu principal ativo, o Meta-Modelo de Descoberta de Conhecimento (em inglês - *Knowledge Discovery Meta-Model* - KDM).

Realizou-se também uma revisão bibliográfica sistemática para investigar as principais técnicas e trabalhos disponíveis na literatura para auxiliar o desenvolvido do projeto em questão. O bolsista elaborou a monografia para o Exame de Qualificação e apresentou a mesma para a banca. Também são descritas nesta seção as publicações e eventos científicos nos quais o bolsista participou diretamente.

3.1 Atividades Técnicas Previstas no Cronograma do Projeto

3.2 Disciplinas Regulares do Curso de Doutorado do ICMC/USP

De acordo com o tema central do projeto, foram escolhidas algumas disciplinas julgadas importantes para o bom aproveitamento do trabalho proposto no cronograma inicial. O quadro de disciplina cursadas pelo bolsista é mostrado a seguir:

1. Engenharia de Software Experimental
    - Carga Horária - 90
    - Créditos: 6 (seis)
    - Conceito Obtido: A

2. Revisão Sistemática em Engenharia de Software

    - Carga Horária - 90
    - Créditos: 6 (seis)
    - Conceito Obtido: A

3. Validação e Teste de Software

    - Carga Horária - 180
    - Créditos: 12 (doze)
    - Conceito Obtido: A

4. Preparação Pedagógica

    - Carga Horária - 60
    - Créditos: 4 (quatro)
    - Conceito Obtido: A

5. Especificação formal de software

    - Carga Horária - 180
    - Créditos: 12 (doze)
    - Conceito Obtido: A

De acordo com as regras do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional do ICMC os alunos devem obter uma quantidade mínima de créditos para realizar determinadas tarefas, tais como, realizar o exame de qualificação e entregar a dissertação/tese. Nesse contexto, vale ressaltar que o bolsista em questão com o comprimento das disciplinas listadas acima obteu-se um total de 40 créditos, sendo apto para realizar o exame de qualificação que exigi o mínimo de 24 créditos.

Maiores informações sobre o desempenho do bolsista em cada uma das disciplinas e os conceitos obtidos podem ser avaliados mais cuidadosamente no histórico escolar deste relatório (ver Apêndice A).

3.3 Estudo sobre Reengenharia de Software

Software é um produto que evolui constantemente para satisfazer às necessidades de seus usuários. Para isso é necessário submetê-lo a constantes atividades de manutenção, que podem degradar o código-fonte, tornando-se cada vez mais difícil de mantê-lo uma vez que, na maioria das vezes, o software não é atulizado, bem como a sua a documentação, culminando em uma situação em que a única documentação confiável é o código-fonte. Sistemas com essas características são denominados sistemas legados. Usualmente, sistemas legados são candidatos à reengenharia.

Segundo Fowler et al (1999) reengenharia no contexto da evolução do software é utilizada para melhorar a qualidade do software, ou seja, melhorar a extensibilidade, modularidade, reusabilidade, complexidade e manutenibilidade. Por exemplo, reengenharia é útil para converter sistemas legados ou códigos deteriorados em unidades mais modularizadas ou até mesmo migrar tais sistemas para diferentes linguagens de programação e paradigmas.

De acordo com Chikofsky e Cross II (1990), a reengenharia de software é necessário para converter o código legado ou deteriorado em um código mais modular e estruturado, ou até mesmo alterar o paradigma de programação. Segundo **?** mais da metade dos projetos que aplicam reengenharia falham ao lidar com desafios específicos. De acordo com esse autor, tanto a carência de padronização durante a atividade de reengenharia quanto a falta de apoio computacional efetivo são os principais problemas que acarretam esse grande número de falhas.

A falta de um processo padronizado de reengenharia usualmente é um problema pois, usualmente tal processo é realizado de forma totalmente *ad hoc*. Além disso, a carência de um apoio computacional efetivo para auxiliar a atividade de reengenharia também é um problema, uma vez que tal atividade não é trivial e requer que várias mudanças sejam realizadas tanto no código-fonte como em outros artefatos, e.g., documentos de requisitos, casos de uso e diagrama de classes.

3.4 Estudo sobre Desenvolvimento Dirigido a Modelos

A proposta do Desenvolvimento Dirigido a Modelos (*Model-Driven Development - MDD*) é reduzir a distância semântica entre o problema do domínio e a solução/implementação. Assim, o engenheiro de software não precisa interagir inteiramente com o código-fonte, podendo-se concentrar em modelos que possuem maiores níveis de abstração. Um mecanismo é responsável por gerar automaticamente o código-fonte por meio dos modelos. No MDD, modelos não apenas guiam as tarefas de desenvolvimento e manutenção, mas são partes integrante do software assim como o código-fonte, servindo como entrada para ferramentas de geração de código reduzindo os esforços dos desenvolvedores **??**

No desenvolvimento tradicional, ou seja, sem seguir o MDD, artefatos de alto nível (ex., modelos, diagramas) usualmente são produzidos antes da codificação e costumam ser úteis apenas nas etapas iniciais do ciclo de desenvolvimento. Conforme o desenvolvimento evolui, mudanças são aplicadas somente no código-fonte e não nos modelos/diagramas. Assim, tais artefatos acabam se tornando incoerentes, ou seja, não refletem o que o código-fonte apresenta, o que faz com que o tempo e os esforços gastos na construção desses artefatos não sejam diretamente aproveitados na produção do software **?**. Contrapartida, o MDD tem o foco nos modelos e busca simplificar o processo de desenvolvimento de software. Modelos são mais intuitivos para representação do conhecimento e menos dependentes do código-fonte, de forma que podem ser reutilizados facilmente em diferentes projetos. Diferentemente, o código-fonte possui uma linguagem que é densa e codificada, tornando-se difícil identificar, extrair e reutilizar o conhecimento apenas pela leitura do mesmo (**?**).

3.5 Estudo sobre Modernização Dirigida à Arquitetura

Reengenharia de Software e MDD são duas abordagens diferentes e que foram pesquisadas separadamente durante anos. Porém, recentemente pesquisadores identificam interesses análogos entre ambas abordagens.

Como dito anteriormente, o principal objetivo da Reengenharia de Software é converter sistemas legados para novos sistemas sem alterar a sua funcionalidade, ou seja, isso implica em entender como o sistema legado foi implementado. Portanto, tal sistema legado deve ser convertido a um nível maior de abstração com o intuito de auxiliar o engenheiro de software a se concentrar apenas em informações importantes desse sistema, nesse ponto que MDD e Reengenharia de Software podem ser utilizados em conjunto.

Nesse contexto, a OMG OMG (*Object Management Group*) propôs a abordagem Modernização Dirigida à Arquitetura (*Architecture-Driven Modernization* - ADM), a qual tem como intuito automatizar e formalizar/padronizar o problema tradicional da Reengenharia de Software, i.e., ADM melhora a abordagem de reengenharia de software tradicional com a utilização de MDD **?**. Segundo **?** ADM é uma padronização definida pela OMG para auxiliar a atividade de reengenharia de sistemas legados, com o diferencial da utilização dos princípios da MDD (ver Seção **??**). ADM difere das abordagens tradicionais de reengenharia de software por dois principais motivos: (*i*) ADM considera todos os artefatos de um sistema legado como modelos e (*ii*) refatorações do sistema legado são realizadas nos modelos e depois gera-se um novo código-fonte refatorado tendo como base tais modelos.

De acordo com **?** ADM tem algumas vantagens quando comparada com abordagens de reengenharia convencionais: (*i*) permite que pesquisadores definam técnicas de refatorações independente de linguagem e plataforma; (*ii*) refatorações podem ser definidas como modelos, assim, podem ser reutilizadas, ou seja, utiliza o metamodelo KDM, o qual tem como um dos objetivos aumentar a interoperabilidade entre as ferramentas de reengenharia e (*iii*) com a abordagem ADM é possível criar técnicas de refatorações genéricas e específicas.

- estudar ferramentas para criar uma Linguagem Específica de Domínio: fez-se necessário identificar e estudar algumas ferramentas que auxiliam o desenvolvimento de Linguagem Específica de Domínio. Foram identificadas as seguintes ferramentas,

XText[4], Eclipse EMF[5] e Eclipse GMF[6]. Tais ferramentas serão estudadas para o desenvolvimento de um Linguagem Específica de Domínio, a qual será utilizada no terceiro passo da abordagem proposta;

- Estudo detalhado da abordagem denominada Arquitetura Dirigida a Modelo - (ADM): conforme descrito no projeto submetido anteriormente pretende-se criar uma abordagem para auxiliar o engenheiro de software durante a atividade de reestruturação de sistemas legados. Dessa forma, vez se necessário estudar realizar a modernização de sistemas legados com a utilizanção de

[4]http://www.eclipse.org/Xtext/
[5]http://www.eclipse.org/modeling/emf/
[6]http://www.eclipse.org/modeling/gmp/

3.6 Organização da Proposta

4 Fundamentação Teórica

4.1 Mineração de Padrões de Projeto

4.2 Mineração de Interesses Transversais

5 Proposta

5.1 Objetivos

5.2 Desafios de Pesquisa com Relação ao Projeto

5.3 Metodologia

5.3.1 Atividades e Cronograma

5.4 Avaliação

5.5 Resultados Esperados

6 Trabalhos Relacionados

Referências

Bisbal, J.; Lawless, D.; Wu, B.; Grimson, J. Legacy information systems: issues and directions. *Software, IEEE*, vol. 16, no. 5, pp. 103 –111, 1999.

Chikofsky, E. J.; Cross II, J. H. Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, vol. 7, no. 1, pp. 13–17, 1990.

Demeyer, S. Refactor Conditionals into Polymorphism: What's the Performance Cost of Introducing Virtual Calls? *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*, 2005.

Demeyer, S.; Du Bois, B.; Verelst, J. Refactoring - Improving Coupling and Cohesion of Existing Code. *Reverse Engineering, 2004. Proceedings. 11th Working Conference on*, 2004.

Fowler, M.; Beck, K.; Brant, J.; Opdyke, W.; Roberts, D. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, 1999.

Kiczales, G.; Lamping, J.; Mendhekar, A.; Maeda, C.; Lopes, C. V.; Loingtier, J.-M.; Irwin, J. Aspect-oriented programming. In: *ECOOP*, 1997, pp. 220–242.

Monteiro, M.; Fernandes, J. Refactoring a Java Code Base to AspectJ - An Illustrative Example. *Proceedings of the 21st IEEE International Conference on Software Maintenance*, 2005.

Tilley, S. Perspectives on legacy system reengineering. (Acessado 09 de Janeiro de 2012), 1995.