Um Método para a Refatoração de Software Baseado em Frameworks de Domínio

Víctor P. A. Barros
Universidade Tecnológica Federal do Paraná
Av Monteiro Lobato, s/n, Km 04
Ponta Grossa, Paraná
victorambiel@outlook.com

Simone N. Matos
Universidade Tecnológica Federal do Paraná
Av Monteiro Lobato, s/n, Km 04
Ponta Grossa, Paraná
snasser@utfpr.edu.br

RESUMO

Este trabalho criou um método de refatoração usando como referência os métodos da literatura, capaz de ajudar os desenvolvedores na refatoração de aplicações construídas com os conceitos de *frameworks* de domínio. O método proposto é formado por três etapas principais: *Entender o sistema*, *Ordenar os módulos* e *Refatorar Módulos*. A diferença entre o método proposto e os da literatura é que prevê a aplicação de metapadrões, inversão de controle e uso de ferramenta de refatoração em suas etapas. O estudo de caso em que o método foi aplicado é o Framework de Formação de Preço de Venda (FrameMK), desenvolvido pelo Grupo de Pesquisa em Sistemas de Informação do Câmpus Ponta Grossa, que tem a finalidade de calcular o preço de venda de um produto ou serviço. Os resultados da aplicação do método no FrameMK foram: melhorou a complexidade do código, diminui a quantidade de *bad smells* (sintomas no código fonte que indicam problemas mais graves no software) e a duplicação de código, o código ficou mais reusável e flexível e houve um aumento na qualidade do software em relação a expectativas do seu ciclo de vida.

Palavras-chave

refatoração; frameworks; metodologias

ABSTRACT

This paper created a method refactoring with reference to the methods of literature, able to assist developers in refactoring applications built with domain frameworks concepts. The proposed method consists of three main steps: Understanding the system, Sort modules and Refactor modules. The difference between the proposed method and the literature is that it provides for metapatterns, inversion of control and use of refactoring tool in their steps. The case study in which the method was applied is the Framework of Sales Price Formation (FrameMK), developed by the Research Group Information Systems on Campus Ponta Grossa, which have the purpose of calculating the selling price of a product or service. The results of applying the method in FrameMK were: improved code complexity, reduces the quantity of bad smells (symptoms in the source code that indicate more serious problems in the software) and the duplication code, the code became more reusable and flexible and there was an increase in the quality of software in relation to expectations of its cycle life.

Keywords

refactoring; frameworks; methodologies

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

1. INTRODUÇÃO

A refatoração de software foi apresentada por Fowler [10] como um conjunto de técnicas que facilita modificar a estrutura interna do software, sem alterar o seu comportamento externo.

O processo de refatoração pode ser melhorado com a adoção de métodos que guiam o processo na identificação das partes do código que devem ser refatoradas e indicam qual a melhor técnica a ser utilizada em determinado tipo de problema.

Alguns métodos de refatoração já foram publicados, como o de Mens e Tourwé [11] que possui uma sequência de seis passos a serem seguidos para se aplicar a refatoração e não é focado em um tipo de software específico. Outro trabalho é o de Rapeli [12], que está focado na refatoração de sistemas em Java com Padrões de Projeto. Estes métodos não exploram a refatoração de aplicações construídas com o conceito de *frameworks* de domínio, as quais contém características que envolvem aplicação de metapadrões, padrões de projeto, inversão de controle, entre outros.

Um *framework* é uma estrutura que tem como o objetivo prover uma funcionalidade genérica, que serve de apoio para a construção de uma outra aplicação [9]. Eles podem ser classificados como de infra-estrutura, *middleware* e domínio (ou aplicação). Os *frameworks* de infra-estrutura simplificam o desenvolvimento de sistemas de infra-estrutura portáveis e eficientes. Os *frameworks middleware* integram aplicações e componentes distribuídos, escondendo o baixo nível de comunicação entre os componentes distribuídos. Os *frameworks* de domínio têm um foco no desenvolvimento de aplicação em domínios específicos tais como: agricultura, formação de preço de venda, entre outros.

Este trabalho criou um método de refatoração que pode ser aplicado na estrutura de software construído com os conceitos de um *framework* de domínio, utilizando como base

métodos já publicados, buscando assim obter um melhor resultado na refatoração do *framework*. Para isto, buscou-se quais os conceitos utilizados para a construção de um *framework*, assim foi possível identificar quais pontos devem ser considerados em sua refatoração, introduzindo os conceitos de metapadões, inversão de controle, padrões de projeto e técnicas de refatoração. Também foi proposto automatizar e facilitar a aplicação da refatoração utilizando ferramentas que auxiliam na análise de código.

O uso do método proposto foi aplicado no *framework* de domínio na Formação de Preço de Venda (FrameMK) [4] que está em desenvolvimento por acadêmicos da computação da Universidade Tecnológica Federal do Paraná, câmpus Ponta Grossa. Este *framework* tem como principal objetivo oferecer um ambiente em que o usuário possa calcular o preço de venda de um produto ou serviço. Foi escolhido este *framework* porque o código da aplicação foi desenvolvido por várias pessoas e com isto necessitava de alterações em sua estrutura interna.

2. METODOLOGIA

O método proposto (Figura 1) é dividido em três etapas principais, sendo que para cada uma foi criado um fluxograma explicitando o seu funcionamento.

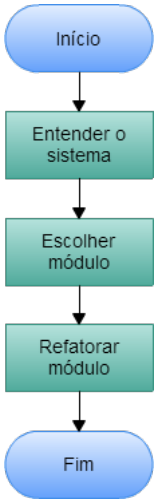

Figura 1: Processo Geral do Método Proposto

A primeira etapa *Entender o sistema* tem a finalidade de permitir ao desenvolvedor um entendimento geral de como o sistema funciona. A segunda etapa, *Escolher módulos*, tem como objetivo classificar os módulos que compõem o sistema para que estes possam ser refatorados na etapa posterior. Por fim, a última etapa *Refatorar módulo* realiza o processo refatoração propriamente dito.

Dentre as etapas do método, a primeira está baseada no método de Rapeli [12], que contém a etapa de *Entender o sistema* e o passo de *Gerar diagramas de classe*. As outras etapas foram determinadas por este trabalho.

2.1 Etapa 1 - Entender o Sistema

Esta etapa consiste em três passos que são essenciais para a realização da refatoração do sistema. O primeiro passo é o *Utilizar o sistema*, onde seu objetivo é que o desenvolvedor tenha o primeiro contato com o sistema que será refatorado, verificando como ele funciona e interage.

O segundo passo, *Verificar módulos*, tem como objetivo identificar quais são os módulos do sistema. No método proposto foi considerado um módulo como sendo um subsistema ou pacote. Muitos sistemas são divididos em subsistemas, como por exemplo, um *Enterprise Resource Planning* (ERP), que possui módulos que se interagem entre si, tais como: Fiscal, Financeiro, entre outros. Separar o sistema em módulos facilita o entendimento sobre seu funcionamento. Muitas vezes a separação por módulos não é explícita, sendo assim, pode-se considerar um módulo um conjunto de pacotes que tem como objetivo fornecer uma funcionalidade específica, como por exemplo, um pacote de Regra de Negócio que contempla todas as classes de regras lógicas do sistema.

O terceiro passo é o *Gerar diagrama de classe dos módulos*. O objetivo é criar o diagrama de classe dos módulos que foram identificados no passo anterior, podendo assim, ter uma representação visual e mais clara de como as classes do sistema funcionam e se relacionam entre si.

2.2 Etapa 2 - Escolher Módulo

Esta etapa do processo de refatoração tem como foco ordenar de forma crescente os módulos pela quantidade de *bad smells*, para que o processo de refatoração se inicie do menor para o maior módulo.

Para se obter a quantidade de *bad smells* de cada módulo é utilizada uma ferramenta de análise de código. O presente trabalho utilizou a ferramenta *SonarQube* [6], pois os pesquisadores já tinham um conhecimento prévio sobre seu funcionamento, porém há outras ferramentas disponíveis que podem ser utilizadas, como a *FindBugs* [3], *Checkstyle* [1] e *CodePro AnalytiX* [2].

A ordenação dos módulos fica a critério do desenvolvedor utilizar o melhor algoritmo de ordenação que se encaixa em seu contexto, desde que o algoritmo utilizado cumpra com a finalidade dessa etapa, ordenando os módulos do menor para o maior, em relação a quantidade de *bad smells* de cada módulo.

A escolha de começar pelo módulo que contém a menor quantidade de *bad smells* pode facilitar a refatoração para desenvolvedores menos experientes, pois o número de refatorações que deverão ser feitas tendem a ser menores.

2.3 Etapa 3 - Refatorar Módulo

A última etapa do método proposto é *Refatorar Módulo* e é o momento em que de fato o código é refatorado.

Esta etapa inicia com a iteração de i começando em 1 até m, sendo que m representa a quantidade total de módulos. Dentro da iteração verifica-se se o desenvolvedor deseja refatorar o módulo i, estando os módulos ordenados em ordem crescente da quantidade de *bad smells*, conforme a etapa 2.

Se o desenvolvedor desejar refatorar o módulo, então é utilizada a ferramenta de análise de código novamente, porém agora o foco é realizar a refatoração do módulo propriamente.

Após a análise do projeto a partir da ferramenta, é verificado os *bad smells* existentes, e tem-se uma condição de verificação. Se há *bad smells* no sistema, o próximo passo é aplicar as técnicas de refatoração, essas focadas em *frameworks*.

Tais técnicas são: as técnicas de refatoração apresentadas

por Fowler [10], aplicação de padrões de projeto, aplicação de metapadrões e aplicação de inversão de controle (IC). A escolha de qual técnica deve ser realizada inicialmente fica a critério do desenvolvedor a medida que ele analisa o código e consegue identificar a aplicação de alguma técnica.

As técnicas de refatoração de Fowler [10] e aplicações de padrões já são conhecidos na literatura. O processo de refatoração usando metapadrões e IoC pode ser encontrado no trabalho publicado por Barros [8].

3. RESULTADOS

Para a refatoração do FrameMK foram modificadas ao todo 30 (trinta) classes dentro do módulo *app*, 23 (vinte e três) classes do módulo *Persistence* e *BussinnesRule* e 5 (cinco) classes do módulo *service*.

Foram aplicadas 295 (duzentos e noventa e cinco) refatorações sugeridas pela ferramenta *SonarQube* [6], sendo removidos 61% dos *bad smells* presentes dentro do módulo *app*.

A partir da ferramenta *SonarQube* [6] também foi possível medir o *SQALE* [7] do módulo *app*, tendo como parâmetros o quão objetivo, preciso, de fácil reprodução e automatizado é o código, sendo que antes da refatoração a nota do módulo *app* no *SQALE* [7] era B e após a refatoração ela subiu para a nota A. A Tabela 1 ilustra como eram algumas características do FrameMK antes e depois da refatoração.

Tabela 1: Estatísticas da refatoração do FrameMK

Informações	Antes	Depois
Quantidade de classes	37	39
Quantidade de bad smells	477	182
Complexidade	531	236
Duplicações	39,2%	35,8%
Linhas de código	2797	2921
Quantidade de funções	144	151
SQALE rating	B	A

Utilizando a técnica de Fowler [10] foi possível aplicar uma refatoração dentro do módulo *app*, que gerou a redução da duplicação de código de 3 (três) classes que continham códigos idênticos.

Foi utilizado também o metapadrão *Unification*, sendo necessário a criação de uma nova classe, assim foi possível unificar 3 (três) métodos utilizados por 3 (três) classes diferentes.

Durante o processo de refatoração foram identificados pontos de maior facilidade para a aplicação do método e pontos de maior dificuldade.

Na primeira etapa *Entender o sistema*, o *framework* de domínio FrameMK já era conhecido pelo autor, assim tinha-se um conhecimento prévio de como interagir com o *framework*, porém em alguns casos o desenvolvedor que irá aplicar a refatoração utilizando o método proposto pode não ter um conhecimento prévio sobre o *framework* que irá refatorar, o que pode levar um pouco mais de tempo para se executar tal etapa. A maior dificuldade da primeira etapa foi realizar a engenharia reversa do código para gerar o diagrama de classes, pois o FrameMK contém 344 classes, e que muitas delas não possuem dentro de suas estruturas os atributos de outras classes as quais estão relacionadas.

A segunda etapa em que os módulos são ordenados também não houve dificuldade, pois foi implementado o algoritmo *Insertion Sort* para ordenar os módulos. Sendo assim, foi necessário apenas analisar o módulo a partir da ferramenta *SonarQube* [6] e utilizar os dados obtidos sobre a quantidade de *bad smells* como entrada no algoritmo de ordenação. Este processo poderia ser integrado com a ferramenta, visto que os dados são obtidos por meio dela. Isto representa uma restrição que deve ser resolvida se o método proposto for automatizado.

A terceira etapa em que o sistema foi refatorado, houve maiores dificuldades em relação as outras, pois era necessário um bom conhecimento sobre as técnicas de refatoração, padrões de projeto, inversão de controle e metapadrões para identificar partes de códigos no *framework* que era necessário aplicar tais conceitos.

4. TRABALHOS RELACIONADOS

Em comparação com os outros trabalhos, o método proposto tenta unir o que há de melhor entre o que Rapeli [12] e Mens e Tourwé [11] podem oferecer no contexto de *frameworks*. Assim como Rapeli [12], o método proposto também oferece uma forma melhor de se entender como o sistema funciona, contendo a etapa de gerar diagrama de classes. Também oferece a aplicação de padrões de projeto de forma específica, como um meio de se refatorar o *framework*.

No método proposto também é possível identificar os tipos de refatoração a serem aplicados, assim como o de Mens e Touwé [11].

A Tabela 2 apresenta a comparação entre o método de Rapeli [12], Mens e Tourwé [11] e o método proposto neste trabalho. Diferente dos métodos de Rapeli [12] e Mens e Tourwé [11], o novo método apresenta a utilização de ferramentas automatizadas, que auxiliam na refatoração do *framework*, identificando mais facilmente pontos do código que necessitam de atenção. Também apresenta a refatoração a partir da utilização de Inversão de Controle e Metapadrões, que são dois conceitos utilizados na criação de um *framework*, logo devem ser levados em consideração na refatoração.

Tabela 2: Comparação entre os métodos de Rapeli [12], Mens e Tourwé [11] e Método Proposto

Características	Rapeli	Mens e Tourwé	Método Proposto
Compreender a funcionalidade do sistema	**X**	**X**	**X**
Gerar diagrama de classe	**X**		**X**
Aplicar padrões de projeto de forma específica	**X**		**X**
Testar sistema refatorado	**X**	**X**	
Refatorar para qualquer linguagem		**X**	
Analisar código para refatorar	**X**	**X**	**X**
Identificar tipo de refatoração a serem aplicadas		**X**	**X**
Utilizar ferramentas automatizadas para análise do código			**X**
Aplicar Inversão de Controle			**X**
Aplicar Metapadrões			**X**

O método proposto não contempla etapas que devem ser realizadas para a fase de teste, somente explicita que a validação deve ser realizada, mas não demonstra como. Devido

a isso, na Tabela 2, a característica *Testar sistema refatorado* foi deixada em branco.

A diferença do método proposto em relação aos métodos da literatura são: uso de ferramentas de análise de código, utilização do guia para a aplicação de metapadrões e inversão de controle. Em relação ao método de Mens e Tourwé [11] e Rapeli [12] possui as seguintes semelhanças: compreender o sistema e analisar código. Comparando com Mens e Touwé [11] o método possui formas de identificar os locais do código que a refatoração será aplicada. Considerando o método de Rapeli [12], a igualdade está nas tarefas de aplicar padrões de projeto e criar o diagrama de classe.

5. CONCLUSÃO

Este trabalho apresentou um novo método para a refatoração, porém diferente dos métodos já publicados, pois foca no contexto de *frameworks* de domínio em que se utilizou as características tais como: padrões de projeto, metapadrões, inversão de controle e ferramentas automatizadas para análise de código, além de utilizar também técnicas de refatoração.

O método proposto é composto por 3 (três) etapas principais: *Entender o sistema*, *Ordenar módulos* e *Refatorar módulo*, sendo que cada etapa contém seus passos a serem seguidos.

Para a validação, o método proposto foi aplicado em um estudo de caso, o *framework* de domínio FrameMK, que é desenvolvido e mantido pelo Grupo de Pesquisa em Sistemas de Informação (GPSI) [5]. A aplicação contemplou todas as etapas e passos propostos, desde o entendimento do sistema, seguindo para a escolha dos módulos até chegar na etapa de refatoração, onde foi aplicada as características identificadas como importantes para um *framework* de domínio.

Com as refatorações efetuadas no FrameMK conseguiu-se diminuir significativamente a quantidade de *bad smells* que estavam presentes no código fonte, conseguindo também aumentar a flexibilidade, diminuir a complexidade, reduzir a duplicação de código, além de fornecer também um código fonte mais legível, possibilitando com que o framework possa ser expandido com maior facilidade.

5.1 Trabalhos Futuros

Os trabalhos futuros que podem ser realizados a partir desta pesquisa são: A identificação de novas características na construção de *frameworks* de domínio que devem ser levados em consideração no processo de refatoração. A aplicação de métricas para avaliar quantitativamente a contribuição do método na refatoração de *frameworks*. Aplicar o método proposto em outros estudos de caso ou módulos. Automatizar o processo de detecção de metapadrões em código fonte. Refinar o modelo de classes do FrameMK de modo a facilitar a engenharia reversa. Definir as etapas que devem ser realizadas na fase de validação do *framework*.

6. REFERÊNCIAS

[1] Checkstyle 6.7. http://checkstyle.sourceforge.net/. Accessado: 05/09/2016.

[2] Codepro analytix. https://marketplace.eclipse.org/content/codepro-analytix. Accessado: 05/09/2016.

[3] Findbugs 3.0.1. http://findbugs.sourceforge.net/. Accessado: 05/09/2016.

[4] Framemk. http://gpes.pg.utfpr.edu.br/framemk/. Accessado: 05/09/2016.

[5] Grupo de pesquisa em sistemas de informação (gpsi). http://gpes.pg.utfpr.edu.br/gpes/. Accessado: 05/09/2016.

[6] Sonarqube 5.1.1. http://www.sonarqube.org/. Accessado: 05/09/2016.

[7] Sqale. http://www.sqale.org/wp-content/uploads/2011/05/SQALE-Method-EN-V0-09.pdf. Accessado: 05/09/2016.

[8] V. P. A. Barros. Um método para refatoração de software baseado em frameworks de domínio, 2015.

[9] M. Fayad, D. C. Schmidt, and R. E. Johnson. *Building application frameworks: object-oriented foundations of framework design*. 1999.

[10] M. Fowler. *Refactoring: improving the design of existing code*. Pearson Education India, 2009.

[11] T. Mens and T. Tourwé. A survey of software refactoring. *IEEE Transactions on software engineering*, 30(2):126–139, 2004.

[12] L. R. C. Rapeli. *Refatoração de sistemas Java utilizando padrões de projeto: um estudo de caso*. PhD thesis, Dissertação (Mestrado em Ciência da Computação), UFSCar–São Carlos, 2005.