Arquitetura de Alto Nível do Framework de Formação de Preço de Venda

Vinícius Camargo Andrade [Bolsista PIBIC/ FUNDAÇÃO ARAUCÁRIA], Simone Nasser Matos [orientadora]

Depto. de Informática
Campus Ponta Grossa
Universidade Tecnológica Federal do Paraná - UTFPR
Av. Monteiro Lobato, s/n - Km 04 CEP 84016-210 - Ponta Grossa - PR - Brasil

vinicius.utfpr@uol.com.br, snasser@utfpr.edu.br

Resumo - Atualmente não há no mercado um software que implemente diversos métodos de formação de preço de venda e que seja gratuito. Conscientes deste problema, o Grupo de Pesquisa em Engenharia de Software da Universidade Tecnológica Federal do Paraná, Campus Ponta Grossa, está desenvolvendo um *framework* que implementa diversos métodos de formação de preço de venda. Neste projeto estudaram-se algumas metodologias de desenvolvimento de *framework* e usou a abordagem dirigida a responsabilidades para criar o modelo de requisitos do *framework* no domínio de preço de venda. O modelo contém a identificação dos pontos de estabilidade e flexibilidade dos métodos de preço de venda, núcleo para o desenvolvimento de *framework*s.

Palavras-chave Metodologias, *Framework* de Domínio, Preço de Venda.

Abstract - Nowadays there is not software in the market that implements various formation methods of selling price that are freeware. Aware of this problem, the Research Group in Software Engineering from the Technological University of Paraná, Ponta Grossa Campus, is developing a framework to implement various methods of formation of sales price. In this project we studied some framework development methodologies and used the directed approach towards the responsibilities to create the framework requirements model in the field of selling price. The model contains the identification of points of stability and flexibility of selling price methods, core to the development of frameworks.

Key-words Methodologies, Domain framework, Sales Price.

INTRODUÇÃO

Segundo Silva (2001), para uma empresa obter sucesso, vários aspectos importantes devem ser levados em consideração, entre eles está o estabelecimento do preço de venda, uma vez que o mercado está cada vez mais competitivo devido às tarifações e impostos, da necessidade de novos investimentos e do surgimento de novas despesas.

Com o crescimento da concorrência e do mercado consumidor, as empresas se obrigaram a procurar novas formas de analisar e tomar decisões que atenda as suas necessidades, além dessas decisões serem satisfatórias, elas precisão ser tomadas com rapidez, pois o mercado é variável e o preço de venda precisa ser revisto a todo o momento.

Pensando nisso, foi proposto um projeto para a criação da arquitetura geral do *framework*, implementando um *subframework* no modelo arquitetural, utilizando ferramentas para a modelagem e desenvolvimento, contendo interface gráfica, aplicando padrões de projeto, metapadrões e os *subframeworks* já obtidos.

METODOLOGIA

O processo de desenvolvimento de *framework*s de domínio corresponde a uma evolução iterativa de sua estrutura de classes, que envolve atividades como identificação de classes,

modelagem de cenários e identificação de estados de objetos, entre outros.

Uma abordagem que pode ser utilizada para o desenvolvimento de *framework*s é a abordagem PDR-DFD (Processo Dirigido a Responsabilidade aplicado ao Desenvolvimento de *Framework* de Domínio). Na PDR-DFD as responsabilidades são declarações gerais sobre as ações que um objeto executa e mantém, e sobre as decisões principais produzidas por um objeto que afeta os outros [2].

Essas responsabilidades são encontradas a partir das ações que um sistema deverá executar e nas informações que necessitam ser mantidas ou criadas [3]. Além disso, a PDR-DFD foi construída com o objetivo de dar um contexto adequado de aplicação do método de identificação antecipada de pontos de estabilidade e de flexibilidade proposto por Matos e Fernandes (2008).

RESULTADOS

A arquitetura geral ou de alto nível do framework, ilustrada na Figura 1, foi dividida em dois pacotes principais: *Base* e *Specific*, os quais contêm as classes que são comuns – *frozen spots* – e específicas – *hot spots* –, respectivamente.

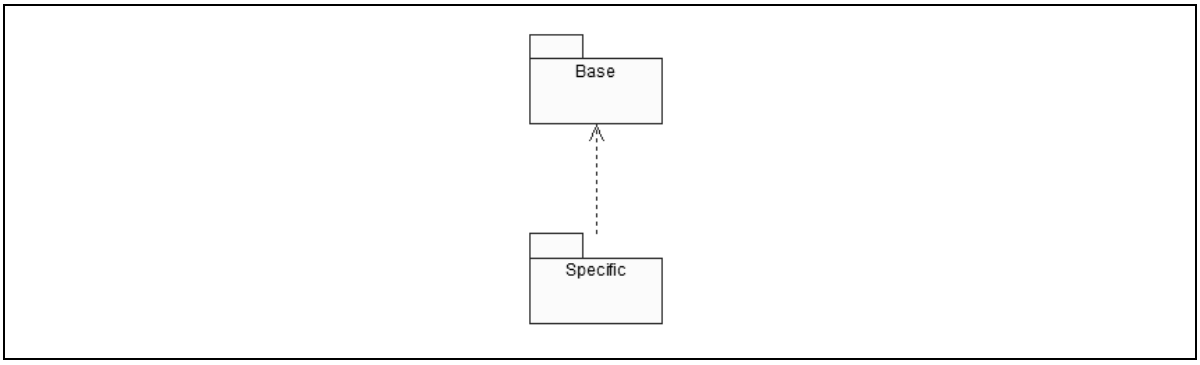

Figura 1. Arquitetura Geral do *Framework*

Há uma relação de dependência entre o pacote "*Specific*" e o pacote "*Base*", pois todos os pontos específicos do *framework* necessitam dos pontos comuns para seu funcionamento.

No interior do pacote "*Base*", verificou-se a necessidade de outros 2 (dois) novos pacotes: *Attributes* e *FoodSystem* (Figura 2). O pacote *Attributes* é responsável por todo o gerenciamento dos atributos utilizados nos métodos de custeio e o pacote *FoodSystem* tem como função gerenciar a entrada de dados que alimenta o sistema afim de efetuar os cálculos necessários para estabelecer o preço de venda do produto.

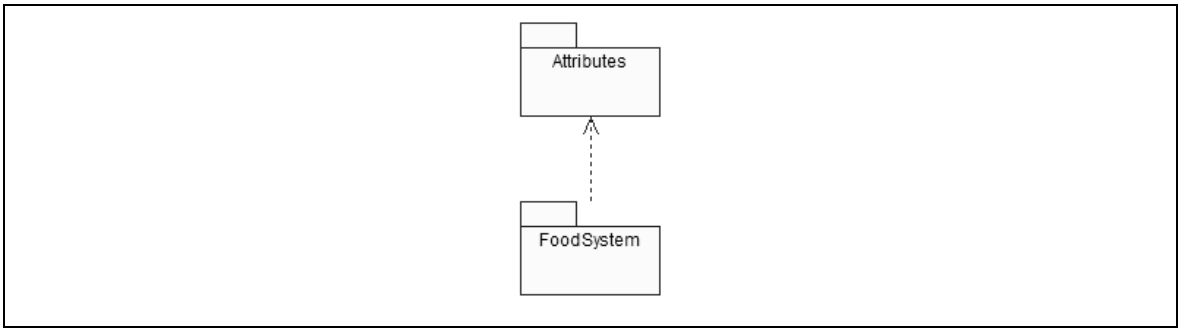

Figura 2. Pacote *Base*

Para alimentar o sistema, ou seja, atribuir valores aos atributos é necessário haver atributos cadastrados, por este motivo há uma relação de dependência entre o pacote *FoodSystem* e o pacote *Attributes*.

As partes específicas, *hot spots*, de todos os *subframeworks* que compõem o *framework* de otimização de Preço de Venda permanecem no interior do pacote *Specific* e é ilustrado pela Figura 3.

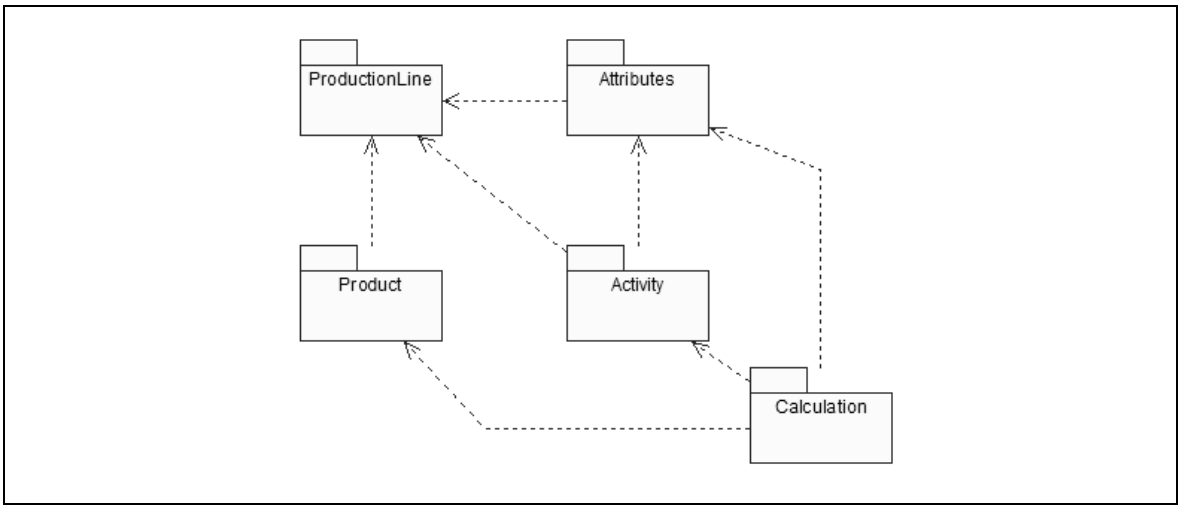

Figura 3. Pacote *Specific*

Alguns *subframeworks* são comuns entre os métodos de formação de preço de venda, como por exemplo, o *subframework Attribute* e *Calculation*, porém a implementação de ambos são diferentes para cada método a ser empregado. O *subframework Attribute*, não contém os mesmos atributos nos 3 (três) métodos estudados, como o *Calculation* também não possui a mesma fórmula para calcular o preço de venda de um determinado produto.

O pacote *ProductionLine* conterá a implementação do *subframework ProductionLine*, que tem como principal função exibir todas as linhas de produção, podendo o usuário cadastrar novas linhas, editar ou desabilitar linhas já existentes. No pacote *Product* será implementado o *subframework Product*, o qual será responsável por exibir todos os produtos cadastrados, podendo o usuário efetuar o cadastro de um novo produto, editá-lo ou desabilitá-lo. No pacote *Activity* será implementado o *subframework Activity*, que terá como função principal exibir todas as atividades existentes, podendo o usuário efetuar o cadastro de uma nova atividade, editá-la ou desabilitá-la. No pacote *Attribute* foi implementado o *subframework* Gerenciar Atributo, sua função é exibir todos os atributos existentes, dando ao usuário a opção de cadastrar, editar ou desabilitar um atributo.

Com exceção do pacote *Attribute*, todos os outros *subframeworks* contidos no pacote *Specific* são em sua totalidade específicos do método ABC.

A relação de dependência existente entre estes *subframeworks* é simples. O *subframework ProductionLine* é o único que não depende de nenhum outro, pois para efetuar o cadastro de uma nova linha de produção é necessário apenas o nome desta nova linha. Já o *subframework Product*, necessita o nome do novo produto e também da linha de produção que ele pertence. O mesmo acontece com o *subframework Attribute*, que precisa do nome do novo atributo a ser cadastrado juntamente com a linha de produção que este produto esteja relacionado. O *subframework Activity* além de necessitar do nome da nova atividade, também se relaciona com um atributo e com uma linha de produção. O *subframework Calculation* é responsável pelo cálculo do preço de venda do produto, porém para obter êxito em sua função, deve estar com todos os atributos e valores disponíveis.

DISCUSSÃO E CONCLUSÕES

Neste artigo apresentaram-se as atividades desenvolvidas durante o período de desenvolvimento do projeto. Realizaram-se as pesquisas bibliográficas do assunto para então desenvolver a arquitetura geral do *framework* e a implementação do *subframework* "Gerenciar Atributo" no modelo arquitetural.

A arquitetura proposta contém identificados os pontos comuns e flexíveis no domínio de preço de venda. Isto facilita o reuso da estrutura bem como sua expansão para um método de preço de venda que não foi analisado inicialmente.

Portanto, a arquitetura proposta atende os requisitos do desenvolvimento de um *framework* que são: reusabilidade, flexibilidade e manutenibilidade.

AGRADECIMENTOS

Agradeço à Fundação Araucária pelo apoio financeiro concedido ao aluno pesquisador e a Universidade Tecnológica Federal do Paraná – Campus Ponta Grossa no que tange ao ambiente de desenvolvimento do sistema.

REFERÊNCIAS

[1] SILVA, C. L. da. Competitividade: mais que um objetivo, uma necessidade. Revista FAE BUSINESS, Blumenau, nº1, Nov 2001.

[2] MATOS, S. N. AND FERNANDES, C. T. Defining the architectural design of frameworks through a group of *subframeworks* created from frozen and hot spots. In: International Conference on Software Engineering Advances, 2006, Tahiti: IEEE Computer Society Press.

[3] WIRFS-BROCK, R. And MCKEAN, A. Object Design: Roles, esponsabilities, and Collaborations, Addison Wesley. 2003.

[4] MATOS, S. N. AND FERNANDES, C. T. Using Responsibilities for Early Identification of Hot Spot Reused in Framework Modeling. In: IEEE International Workshop on Security, Trust, and Privacy for Software Applications. 3. Turku: IEEE Computer Society Press. 2008.

[5] COGAN, S. Activy-Based Costing (ABC) – a ponderosa estratégia empresarial. São Paulo: Pioneira, 1995.

[6] SILVESTRE, W. C. Sistema de custos ABC: uma visão avançada para tecnologia de informação e avaliação de desempenho. São Paulo: Atlas, 2002.

[7] MARTINS, E. Contabilidade de Custo. 6ª Ed., São Paulo: Atlas, 2003. P.220.

[8] SEBRAE. Cálculo do preço de venda. Disponível em <http://www.sebraepr.com.br>. Acessado em: 23 out de 2009.

[9] CRAZUSKI, A; FEITOSA, L. B; CORDEIRO, T, L. Identificação dos pontos de estabilidade e de flexibilidade dos métodos para o estabelecimento de preço de venda. Ponta Grossa, PR, 2008. 66 f. Trabalho de Conclusão de Curso (Graduação) - UTFPR. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Ponta Grossa, 2008.