PROJECT FRAMEWORK MODEL – PMF: PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO ORIENTADOS A OBJETOS

Leandro Doutor Branquinho
Sérgio Teixeira de Carvalho

PROJECT FRAMEWORK MODEL - PMF: UM FRAMEWORK PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO ORIENTADOS A OBJETOS

Leandro Doutor Branquinho[1]
Sérgio Teixeira de Carvalho[2]

Resumo

O paradigma de orientação a objetos, quando aplicado ao desenvolvimento de sistemas de informação, oferece recursos que permitem a construção do *software* por partes, análogo ao *hardware*. Cada parte corresponde a um objeto, que é o responsável e especialista em um serviço no escopo do Sistema de Informação. O paradigma de orientação a objetos permite o desenvolvimento de sistemas de forma mais organizada e com mais qualidade. Esse artigo apresenta o PMF – Projeto Modelo *Framework*, um *framework* criado para auxiliar no desenvolvimento de sistemas de informação orientados a objeto, mostrando sua estruturação de classes, funcionamento, utilização e benefícios, bem como sua importância na instituição onde foi desenvolvido e está sendo utilizado. O artigo apresenta ainda fundamentos de orientação a objetos e *frameworks.*

Palavras–chave: componente, padronização, reusable.

PROJECT FRAMEWORK MODEL - PFM: A FRAMEWORK FOR DEVELOPMENT OF OBJECT-ORIENTED INFORMATION SYSTEMS

Abstract

The object-oriented paradigm, when applied to development of information systems, offers resources that allow the construction of software for parts, analogous to the hardware. Each part corresponds to an object that is the responsible and the specialist in a service in the context of the information system. The oriented-object paradigm allows the development of systems more well organized with more quality. This article presents the PFM - Project Framework Model, a framework created to assist in the object-oriented information systems development, showing its classes, functioning, use and benefits, as well as its importance in the institution where it was developed and is being used. The article still presents a overview about object-oriented paradigm and frameworks.

Key-words: component, padronization, reuse.

[1] Especialista em Orientação a Objetos e Internet pelo Uni-ANHANGÜERA; Analista de Sistemas na UniEvangélica. E-mail: ldb@terra.com.br.

[2] Mestre em Computação pela Universidade Federal Fluminense (UFF),; Professor no Uni-ANHANGÜERA. E-mail: sergiocarvalho@anhanguera.edu.br

Introdução

Desenvolver Sistemas de Informação (SI) em menos tempo e com mais qualidade é um objetivo almejado por profissionais e empresas de desenvolvimento de sistemas de software. O paradigma de Orientação a Objetos oferece recursos para aprimorar o desenvolvimento e tratar questões que dificultam a obtenção de bons resultados no desenvolvimento de SI, como produtividade, padronização e qualidade, dentre outros.

Esse artigo apresenta o PMF[3] – Projeto Modelo *Framework* - um *framework* orientado a objetos que auxilia no desenvolvimento de SI. O artigo descreve primeiramente os fundamentos do paradigma de Orientação a Objetos e de *frameworks*. Em seguida, o PMF é apresentado, mostrando sua estruturação de classes, funcionamento, utilização e benefícios.

Orientação a Objetos

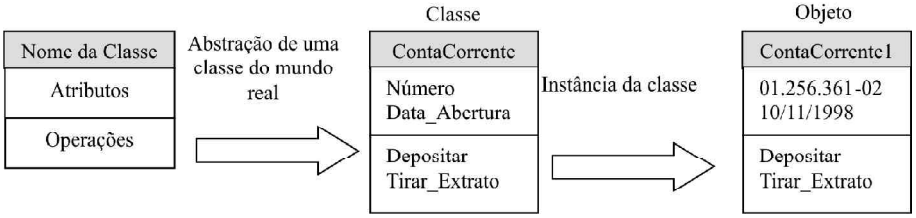
A Orientação a Objetos é um paradigma que foca a construção de Sistemas de Informação a partir de elementos chamados objetos. Os objetos são partes menores do SI especializadas em determinados serviços, representando algo do mundo real como pessoas, veículos ou departamentos de uma empresa. Com ênfase nas informações em vez das ações, facilita o entendimento do problema abordado, tanto de analistas e programadores, quanto de usuários, por existir uma maior familiaridade com os objetos que pertencem ao mundo real.

Um dos fundamentos do paradigma de Orientação a Objetos é a abstração. Abstração significa a capacidade de definir e usar estruturas ou operações complicadas de uma maneira que permita ignorar muitos dos detalhes (SEBESTA, 2003). Sempre se observa coisas e faz-se considerações e analogias relacionadas a elas de acordo com o propósito que nos é relevante. A abstração é o mecanismo usado para o entendimento da complexidade de determinado problema, descartando-se questões consideradas irrelevantes no contexto do

[3] O PMF foi criado pelo primeiro autor.

problema. Dessa forma, voltam-se os esforços e concentração para o que é importante para o entendimento do domínio da solução.

Dados abstraídos são organizados em uma estrutura para representar informações específicas do mundo real. Esta estrutura é chamada classe e especifica o estado (por meio de suas propriedades ou atributos) e o comportamento (por intermédio de seus métodos ou serviços) de objetos. A classe não mantém informações, é apenas um molde para que a partir de determinada classe, seja possível instanciar um ou mais objetos. Composta de três partes - nome, definição de atributos e definição de métodos - a classe pode ser graficamente representada segundo a Figura 1.



Fonte: (BOOCH, 1999).
Figura 1 – Estrutura, abstração e uma instância de uma classe.

As classes podem se relacionar por meio de um mecanismo chamado herança, no qual uma classe mais especializada (subclasse) herda a estrutura da classe mais geral (superclasse), à qual está subordinada na hierarquia. A herança proporciona a reutilização de atributos e comportamento entre as classes envolvidas. A herança pode ser simples quando uma classe é derivada de apenas uma superclasse, e múltipla, quando uma classe é derivada de mais de uma superclasse. Além de subclasse e superclasse, uma classe pode ser do tipo “Final” quando não permite a criação de subclasses e “Abstrata” quando não permite a instanciação de objetos.

Uma vez definidas as classes, objetos podem ser instanciados. Enquanto a classe define a estrutura, o objeto mantém informações em seus atributos.

Estas informações representam propriedades atribuídas a um único objeto, diferenciando-o de outro e definindo o seu estado. Valores de atributos podem ser definidos ou alterados pelos métodos, os quais são funções representando os serviços que irá prestar a outro objeto. Um método manipula os atributos do próprio objeto e não possui acesso direto aos atributos de outro objeto.

A identificação do método juntamente com os tipos dos parâmetros (caso existam) necessários à sua invocação (chamada) formam a assinatura do método. Aos métodos de um objeto podem ser aplicados um mecanismo chamado polimorfismo, uma extensão ou uma adaptação do comportamento genérico do método, permitindo que objetos diferentes respondam a métodos de mesma assinatura, mas com implementações diferentes para um mesmo método, desde que seja mantida a sua finalidade.

Um objeto é visível por outro por meio de sua interface. Nela estão contidos os métodos e atributos que o objeto disponibiliza para ser utilizado. Atributos e métodos podem ou não fazer parte da interface do objeto de acordo com o controle de acesso escolhido: i) *public*: acessível e visível por outro objeto para utilização; ii) *protected*: acessível e visível somente pelas subclasses; iii) *private*: acessível e visível somente pela própria classe.

Determinado objeto é manipulado pela sua interface, comportando-se como uma caixa-preta, aumentando a abstração e a segurança de suas informações. Esta propriedade garante o encapsulamento e o ocultamento de informações (*information hidden*).

Os objetos se comunicam por suas interfaces, trocando mensagens. Uma mensagem é a invocação de um método junto a um objeto para que este processe algo e devolva ao objeto solicitante o resultado esperado.

A Orientação a Objetos propõe a criação de classes a partir de informações abstraídas. A instanciação de objetos e o relacionamento entre estes objetos permitem que os SI sejam montados. Os objetos são reutilizados diminuindo-se o tempo de desenvolvimento. Como são encapsulados, torna-se necessário analisar a interface dos objetos, dispensando considerações sobre sua implementação interna, possibilitando concentrar-se em outros aspectos importantes. O resultado é um produto de melhor qualidade. Assim, o paradigma

de Orientação a Objetos fortalece e estende as possibilidades de construção dos SI em componentes reutilizáveis (componentização).

Frameworks : Uma Abordagem de Reutilização

Um *framework* é um conjunto de classes que cooperam entre si no intuito de atender a questões comuns de um determinado domínio no desenvolvimento de SI (Budd, 2001). Um *framework* oferece uma estrutura encapsulada e genérica de soluções, necessária para o desenvolvimento de aplicações específicas, nas quais características comuns a SI se encontram escritas, implementadas, depuradas, testadas e disponíveis para utilização. Dessa forma, um *framework* pode ser um ponto de partida, uma base, para o desenvolvimento de SI orientados a objeto.

A proposta de um *framework* é permitir a reutilização em nível de projeto, algo mais amplo que a reutilização de classes e objetos. Dessa forma espera-se a diminuição do tempo de desenvolvimento de um SI e melhoria de sua qualidade (FAYAD, 1997).

Ao se utilizar um *framework* o fluxo de controle da aplicação passa a ser gerenciado por ele. O desenvolvedor não pode modificar a sua estrutura interna, uma vez que está encapsulada. Um *framework* é utilizado definindo-se novas classes, com a sobreposição e implementação de novos métodos, e ainda, configurando-se os objetos disponíveis no *framework*, conforme necessário e permitido.

As características comuns aos SI tratadas em um *framework* podem ser representadas como uma interseção, conforme pode ser observado pela Figura 2.

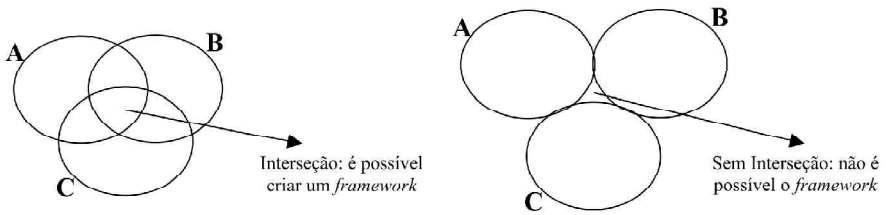


Figura 2 – Região onde é possível a criação de um *framework*

O fato de *frameworks* serem arquiteturas de colaboração entre classes propiciadas pela implementação de funcionalidades comuns entre SI, o diferencia, tanto das Bibliotecas de Classes, onde as classes são independentes, quanto dos *Design Patterns* que não possuem implementação (GAMMA, 1995).

O Framework PMF

A partir da observação de certas tarefas que compreendem a fase de implementação de SI, verificou-se que estas se repetiam bastante, tanto para SI em desenvolvimento quanto para novos SI. Essas tarefas, embora triviais, ainda consumiam um tempo significativo do desenvolvimento. Os transtornos causados por tais tarefas ocasionavam demora na construção de SI e falhas de funcionamento dos SI, decorrentes (i) da falta de concentração do programador, que se envolvia muito e se desgastava com as tarefas repetitivas; (ii) das dificuldades em se manter uma padronização dos SI desenvolvidos; (iii) da pouca ou quase nenhuma reutilização.

No interesse de auxiliar na solução e minimizar esses problemas e proporcionar melhorias no desenvolvimento de SI, foi desenvolvido o Projeto Modelo *Framework* (PMF), uma combinação dos serviços de um *framework* com os mecanismos da Orientação a Objetos.

Construído em *Borland Delphi* (BORLAND, 2004) e utilizado para a construção de SI nesse mesmo ambiente, o PMF oferece vantagens típicas de um *framework*, como a padronização, a extensibilidade (FAYAD, 1997), e a diminuição do tempo de desenvolvimento, aliadas aos recursos de Orientação a Objetos, em especial, a componentização. Tais recursos e vantagens favorecem a construção de SI padronizados e em menos tempo, além de refletir consideravelmente na qualidade, uma vez que possibilita ao programador maior concentração nas regras de negócio do SI em desenvolvimento, diminuindo assim o índice de erros na fase de implementação.

Para a construção do PMF foram observados e analisados os momentos do desenvolvimento: criação de formulários de cadastro, validação de informações digitadas pelos usuários, criação de janelas de consultas, criação de janelas de

parâmetros para emissão de relatórios, tratamento de exceções, controle de acesso e permissão de usuários. Para situações onde o programador se envolvia com tarefas repetitivas e cansativas que consumiam muito tempo no desenvolvimento, foi proposta uma solução única que, combinada à Orientação a Objetos, possibilitou a construção de SI com mais rapidez e mais qualidade.

O PMF é formado por classes que possibilitam a manipulação de informações de bancos de dados juntamente com funcionalidades comuns na implementação de SI. As classes que o compõem possuem as seguintes responsabilidades:

- *Classes de Janelas de Persistência*: os objetos dessas classes são janelas de cadastros de informações que possuem serviços de persistência (incluir, excluir, alterar) e funcionalidades de interface de usuário. Estas classes podem ser herdadas para a especialização de outros de tipos de janelas (janelas de manutenção de clientes, produtos, pedidos de vendas) estendendo a funcionalidade do *framework*. Com objetos dessas classes à disposição, o programador não precisa criar janelas de cadastros a partir do zero.
- *Classes de Janelas de Consulta*: os objetos dessas classes são janelas de consulta à base de dados, oferecendo ao usuário uma interface de consulta amigável e padronizada. Com esses objetos à disposição, o programador se isenta da responsabilidade de criar janelas de consulta para atender às mais diversas necessidades do usuário.
- *Classes de Janelas de Parâmetros de Relatórios*: os objetos dessas classes são janelas de parâmetros de relatórios. Permitem a definição dos campos das entidades do banco de dados que serão disponibilizados na janela para emissão de relatórios pelos usuários.
- *Classes de Persistência*: os objetos dessas classes são usados no mapeamento de objetos para bancos de dados relacionais (REEM, 1999). A partir destas classes podem ser criados objetos que irão manipular a manutenção de entidades no banco de dados. São utilizadas em conjunto com as Classes de Janelas de Persistência, principalmente em janelas que envolvem a manipulação de mais de uma entidade do banco de dados.
- *Classes de Validação de Dados de Entrada*: os objetos dessas classes

verificam a informação digitada em controles existentes nas Janelas de Persistência. Faz validação, tanto no tipo de dado, como números, letras e caracteres especiais, quanto na semântica da informação como CPF, CNPJ, CEP, telefone etc.

O PMF, em execução, instancia Objetos de Janelas de Persistência. Esses objetos associam-se a uma entidade no banco de dados e permitem o controle sobre ela. A esses objetos são inseridos os Objetos de Validação de Entrada de Dados e os Objetos de Persistência. Os objetos de Janelas de Consulta são utilizados quando existe a necessidade de se realizar uma consulta a determina entidade do banco de dados. Para a definição de relatórios, utiliza-se o objeto Parâmetro de Relatório permitindo que sejam definidos campos para que o usuário escolha e defina parâmetros para emissão do relatório. Por fim, os usuários são definidos e para cada um, definidos a sua permissão e acesso por módulo do sistema.

Algumas outras funcionalidades disponíveis no PMF:

- *Controle de Usuários*: uma estrutura de controle de acesso e permissão de usuários possibilita que, para cada módulo do sistema (Janela, Invocação de Métodos de Objetos, Menus), possa ser definido o acesso e a permissão específica por usuário, a qual pode ser limitada desde somente leitura até o controle total no módulo em questão.
- *Configuração de Menus*: o menu principal do SI é configurado automaticamente quando da identificação do usuário pela janela de *login*. Esta é uma funcionalidade disponível, tanto nos SI construídos com base no PMF, quanto no próprio PMF.
- *Métodos de Controle de Transações*: métodos que correspondem a serviços de iniciar, confirmar, cancelar e verificar a situação de transações no banco de dados.
- *Tratamento de Exceções*: as exceções que não são tratadas na parte específica dos SI são gerenciadas pelo PMF. Mensagens de erro para cada exceção são gravadas em um banco de dados para a posterior tradução para o português.
- *Verificação de Informações para Campos Obrigatórios*: para os

campos que são obrigatórios, o PMF exige que o usuário informe o seu valor. Os campos obrigatórios são definidos pelo programador.

- *Informações do Sistema*: o PMF disponibiliza informações do sistema como data e hora (local e do servidor), contagem de registros e informações do sistema operacional.
- *Controle de Objetos Visuais*: controle de ativação de objetos de inclusão de informações e de manipulação do sistema.

PMF: Utilização e Benefícios

O PMF foi desenvolvido em uma Instituição de Ensino Superior, a UniEvangélica: Centro Universitário. Situada na cidade de Anápolis, Estado de Goiás, possui aproximadamente 4300 alunos. Desde a disponibilização do PMF, há cerca de um ano, todos os SI para *desktop* vem sendo implementados a partir de sua plataforma e outros que já se encontravam em operação na instituição foram migrados. Hoje, na instituição, não se cogita a possibilidade da criação de novos SI para *desktop* sem a utilização do PMF, comprovando sua eficácia.

Desde que o PMF foi disponibilizado, foram desenvolvidos vários SI para suprir a necessidade a diversos departamentos do Centro Universitário. Os SI criados estão operando em aproximadamente setenta computadores e estão sendo operados por aproximadamente oitenta usuários. Um número significativo, considerando que o PMF tem pouco mais de um ano de utilização.

A título de consolidação da importância e benefício do *framework*, segue-se a relação de alguns dos SI criados com o PMF e em operação no Centro Universitário:

- SAMPA- Controle de processos judiciais. Utilizado pelo Curso de Direito.
- SICOM - Controle de Mensalidades. Está sendo utilizado na Clínica Odontológica de Ensino do curso de Odontologia, nos esportes de musculação, judô, natação, e na Clínica de Fisiologia do curso de Educação Física.
- Vestibular - Gerência do concurso vestibular da instituição.

- SAI - Sistema de Avaliação Institucional. Utilizado pela assessoria acadêmica.
- PPP - Perfil Profissiográfico Previdenciário. Utilizado pelo departamento de pessoal.
- SMA - Sistema de Matrícula dos Acadêmicos.

Dentre os benefícios obtidos com a utilização do PMF, destacam-se:

- Produtividade: como tarefas básicas de implementação já se encontram prontas no PMF, apenas funcionalidades específicas do SI precisam ser implementadas.
- Reutilização: com a reutilização é possível atender algumas especialidades já tratadas por outros SI.
- Qualidade: como o programador tem à sua disposição soluções para problemas recorrentes e simples, ele pode se concentrar melhor nas regras de negócio e funcionalidades de interface com o usuário do SI, diminuindo a margem de erros e falhas na implementação.
- Padronização: como os SI são criados a partir do PMF, possuem a mesma interface com o usuário, facilitando o treinamento na implantação de novos SI.

Conclusão

Este artigo apresentou o Projeto Modelo Framework – PMF, um *framework* orientado a objetos, e os fundamentos do paradigma de Orientação a Objetos, destacando sua importância no contexto do desenvolvimento de Sistemas de Informação e enfatizando as abordagens de reutilização e construção de sistemas baseada em componentes.

O PMF permite aos desenvolvedores de Sistemas de Informação o emprego de forma mais efetiva das propriedades do paradigma de Orientação a Objetos. Dois benefícios se destacam ao se utilizar o PMF no desenvolvimento de SI: a produtividade, advinda da rapidez na implementação de sistemas, e a qualidade, apresentada na forma de sistemas com menos falhas e mais próximos da satisfação do usuário.

Um *framework* orientado a objeto pode não ser a “bala de prata”

(BROOKS, 1987) para tratar as dificuldades no desenvolvimento de SI. Entretanto, a concepção e a utilização de *frameworks* orientados a objeto tendem a minimizar estas dificuldades e permitir que a construção de SI se assemelhe à construção de *hardware*: modular, baseada em componentes e de fácil integração.

Referências Bibliográficas

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The unified modeling language user guide.** Object Technology Series, Addison-Wesley, 1999.

BORLAND, USA. **Delphi.** Disponível em http://www.borland.com/delphi_net. Acesso: 17 jun. 2004.

BROOKS, F. P. No silver bullet: essence and accidents of software engineering. **Computer Magazine**, v. 20, n. 4, p. 10-19, 1997.

BUDD, T. **An introduction to object-oriented programming**. 3.ed. Addison-Wesley, 2001.

FAYAD, M.; SCHMIDT, D. C. **Special issue on object-oriented application frameworks. Communications of the ACM, v. 40, n.10, 1997.**

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design patterns**: elements of reusable object-oriented software. Addison-Wesley, 1995.

REEM, A. Mapping objects to relational databases, 1999, Disponível em http://www.cs.colorado.edu/~getrich/Classes/csci5817/Term_Papers/reem. Acesso em 17 jun. 2004.

SEBESTA, R. W. **Conceitos de linguagens de programação.** 5.ed. Porto Alegre: Editora Bookman Companhia, 2003.