Frameworks

Pasteur Ottoni de Miranda Junior

1-Definição

Apesar do avanço das técnicas de desenvolvimento de software, a construção de software ainda é um processo extremamente complexo.A reutilização tem sido uma forma de aumento de qualidade e produtividade e a experiência passada norteia a reutilização.
A orientação a objetos é poderosa no que tange à reutilização.Uma base arquitetural para a composição de componentes reutilizáveis auxiliaria muitos dos problemas da construção de software. O benefício básico da reutilização de software é uma maior produtividade. A reutilização também produz uma melhor qualidade, pois o componente de software reutilizável sempre requer mais testes e garantia de qualidade, simplesmente porque as conseqüências de um erro são bem mais sérias e o uso contínuo ocasiona uma maior probabilidade de detecção de erros.
Uma das propostas de reutilização bastante aceita é a de se usar *frameworks* para domínios específicos, que poderiam ser instanciados para produzir novos produtos no domínio. Restrições de custo e tempo impostas ao desenvolvimento moderno de software obrigam os desenvolvedores a abandonar a prática de se partir do zero. Assim sendo, os desenvolvedores devem aderir a uma abordagem que suporte a reutilização, adotando soluções comprovadas, como componentes ,*frameworks* e padrões de projeto.
*O framework (*ou *arcabouço, termo muito usado em português)* é uma técnica da orientação a objetos voltada para a reutilização, que se beneficia de três características das linguagens de programação orientadas a objeto: abstração de dados, polimorfismo e herança. Ele descreve a arquitetura de um sistema orientado a objeto, os tipos de objetos e as interações entre os mesmos. Trata-se em essência de um esqueleto de uma aplicação que pode ser customizado por um desenvolvedor na construção de um software.Um *framework* modela genericamente uma família de aplicativos semelhantes, permitindo uma maior agilidade que se traduz em uma redução de custos no processo de desenvolvimento de software.(Fayad, 1999)
Com *frameworks* não se busca apenas reutilizar simples componentes de software, mas subsistemas, aumentando assim o grau de reutilização e contribuindo para uma melhor qualidade de software.

2-Conceitos

Frozen Spots: definem a arquitetura global de um sistema, seus componentes básicos e os relacionamentos entre eles, permanecendo inalterados – daí o termo congelado - em qualquer instanciação do framework.
Hot Spots: pontos de refinamentos predefinidos, onde a especialização e a adaptação ocorrem. A especialização de hot spots requer a definição de classes adicionais de maneira a sobrepor métodos e/ou configurações de objetos baseados nos componentes já fornecidos pelo framework
Um esquema de *hot spots* contém as seguintes partes (Schmidt, 1997):

- Uma classe abstrata de base, que define a interface para as responsabilidades comuns;

- Classes concretas derivadas, representando cada uma das diferentes alternativas para os aspectos variáveis;
- Parte opcional com relacionamentos e classes adicionais.

Os benefícios advindos da utilização de frameworks são os seguintes
- Modularização
- Reutilização
- Extensão de interfaces
- *Framework* controla a estrutura e o fluxo de controle dos programas.

3-A UML-F (Fontoura,2000)

Para modelar *frameworks,* é preciso projetar *frozen spots* e *hot spots*. Os primeiros podem ser modelados com os recursos que a UML oferece. Quanto aos *hot spots,* são necessárias extensões na UML para a sua adequada especificação.A UML provê os seguintes mecanismos de extensão: estereótipos, *tag values* e restrições.

Os *estereótipos* são mecanismos para permitir a adicionar novos blocos de construção semelhantes aos existentes, mas específicos a um determinado problema. Um estereótipo é representado graficamente como um nome entre os sinais << e >> (ex.: *<<include>>,<<implementation>>)* colocado acima do nome de outro elemento.(Amaral, 2002)

Os *tag values* são meios para proporcionar a criação de novas propriedades, permitindo a criação de novas informações na especificação desse elemento.Um *tag value* é representado como uma seqüência de caracteres entre chaves, colocado abaixo do nome de outro elemento (ex.: {quantidade = 3}). Essa seqüência de caracteres usualmente inclui um nome, um separador (o símbolo =) e um valor.(Amaral, 2002)

As *restrições* são mecanismos para especificar uma nova semântica de algum elemento UML. A restrição é representada como uma seqüência de caracteres entre chaves, colocada próxima ao elemento associado.(Amaral, 2002)

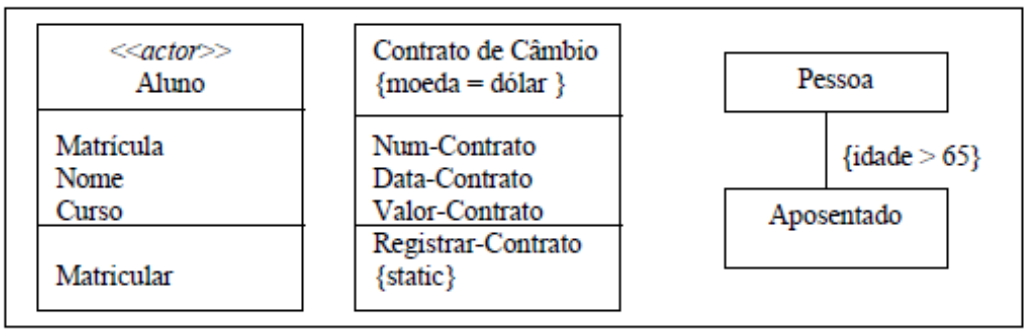

Figura 1-Mecanismos de extensão da UML- (Extraído de Amaral, 2002)

UML-F trata três tipos de *hot spots* em um *framework* orientado a objeto:

- Métodos variáveis: são métodos que têm uma interface bem definida, mas que podem variar em sua implementação de acordo com a instanciação do *framework.*
- Classes estendidas: são classes que podem ser estendidas durante a instanciação do *framework,* recebendo novos métodos, por exemplo.
- Interfaces estendidas ou classes abstratas: permitem a criação de subclasses concretas na instanciação do *framework.* Estas novas classes são chamadas de classes de aplicação, que existem somente quando um *framework* é instanciado.

Os três tipos de *hot spots* anteriores podem ser estáticos (instanciação não requerida em tempo de execução) ou dinâmicos (instanciação requerida em tempo de execução).

As Tags utilizadas em diagramas UML-F são as seguintes:

- *Variable* é aplicado aos métodos de uma classe e representa o *hot spot* métodos variáveis. Indica que a implementação do método varia de acordo com a instanciação do *framework*
- *Extensible* é aplicado a classe e representa o *hot spot* classes estendidas. Indica que sua interface pode ser estendida durante a instanciação do *framework* pela adição de novos métodos.
- *Incomplete* representa o *hot spot* interfaces estendidas. Indica que novas subclasses podem ser criadas nas instâncias do *framework.*
- A *tag appl-class* complementa a definição de classes estendidas. Ela é usada para indicar um aditivo na estrutura de um *framework* onde classes específicas da aplicação podem ser ou já foram adicionadas.
- Os *tags static e dynamic* complementam a notação do *hot spot*, indicando se o mesmo requer instanciação em tempo de execução ou não.

4-Um exemplo

Seja um sistema que exibe informações de cursos a estudantes. Um método *showcourse()* é responsável por controlar o fluxo da aplicação: ele chama *selectCourse()* para seleção do curso e *showContent()* para apresentar o conteúdo do curso selecionado

O método *selectCourse()* é um ponto de variação do *framework,* pois pode ter diferentes implementações em diferentes aplicações web criados com o *framework.* Exemplos de mecanismos de seleção possíveis de serem implementados nesse método: requerer login do estudante, mostrar uma lista e cursos disponíveis ou apenas os disponíveis ao estudante, exibir uma pré-visualização do curso, etc.

Existem 3 categorias de usuários especializados de uma classe Actor. Esta classe pode ter novas subclasses que vão depender da instância do framework

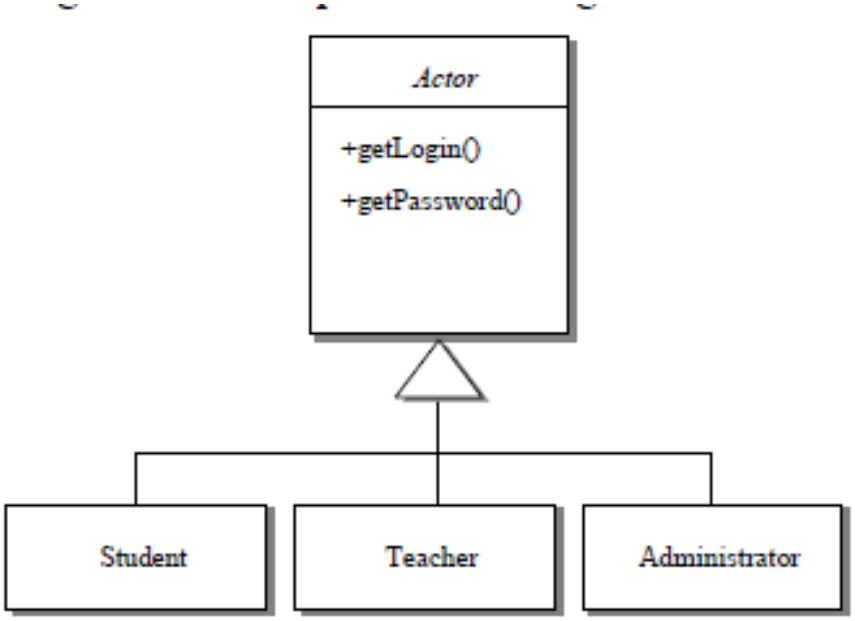

Figura 2-Especializações da classe Actor (Extraído de Amaral, 2002)

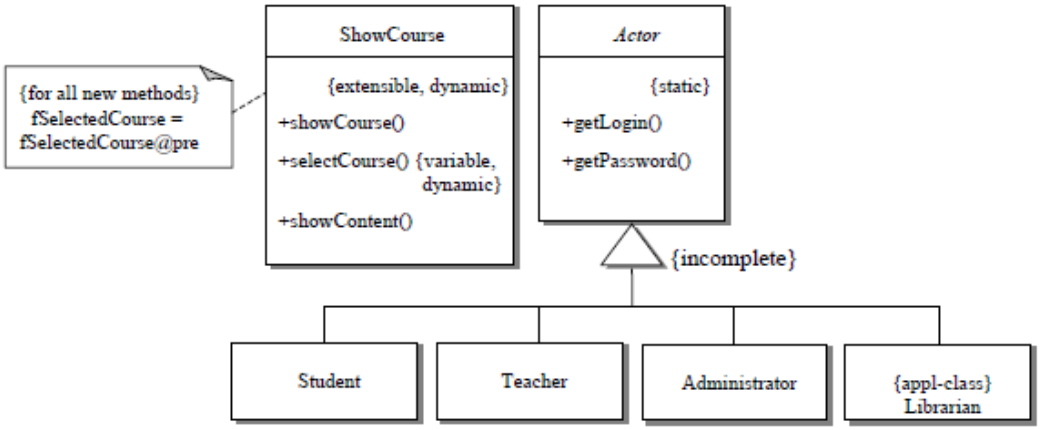

Figura 3- Framework proposto no exemplo (Extraído de Amaral, 2002)

Na Figura 3 acima temos as o diagrama UML-F do *framework* proposto no exemplo. Perceba que a classe ShowCourse é *extensible,* significando que novos métodos podem ser a ela acrescidos quando da instanciação do *framework.* Perceba também que a classe *Actor,* contém a tab {incomplete} significando que, na instanciação, novas subclasses podem ser incluídas. Repare na tag {*variable*} colocada ao lado do método *selectCourse()*, significando que ele pode ter sua implementação modificada na instância.

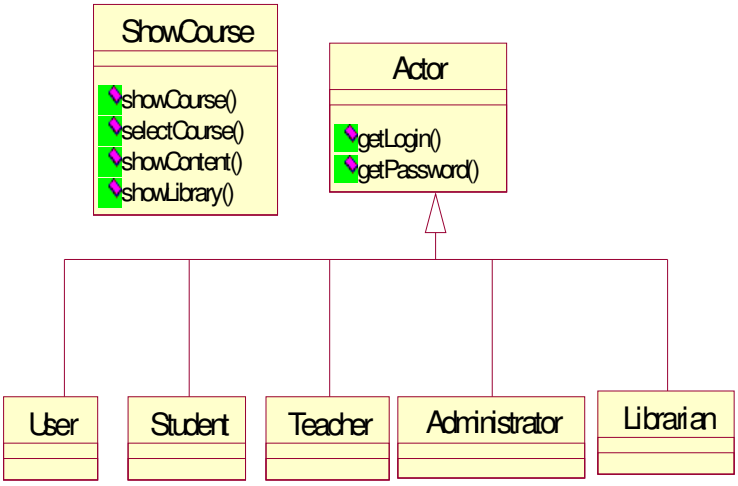

Figura 4- Instância do framework exibido na Figura 3.

Na Figura 4 temos a visualização de uma instância do *framework* mostrado na Figura 3. Perceba que na classe ShowCourse que é *extensible,* foi adicionado o método *showLibrary()*. Já para a classe Actor, que é {incomplete}, adicionamos a classe User. Repare também, que no diagrama da instância do *framework* as tags exibidas no *framework* não são exibidas.
Finalmente, é importante destacar que o diagrama do *framework* nunca é usado para implementação da aplicação. Essa é feita utilizando-se o diagrama da instância do *framework,* como o da Figura 4.

Bibliografia

AMARAL, J. I*ntercâmbio de Frameworks especificados em UML-F através de estruturas XML para apoiar o desenvolvimento de software*. Diss. Mestrado, PUC Minas, 2002.

FAYAD, Mohamed; SCHMIDT, Douglas; JOHNSON, Ralph. *Building Applications Frameworks.* John Willey & Sons,1999.

FONTOURA, Marcus; PREE, Wolfgang; RUMPE, Bernhard. *UML-F: A Modeling Language for Object-Oriented Frameworks*. 14th European Conference on Object Oriented Programming (ECOOP 2000), Lecture Notes in Computer Science 1850, Springer, 63-82, Cannes, França, 2000

PREE, Wolfgang; POMBERGER, Gustav; SCHPPERT, Albert;SOMMERLAND, Peter. *Active Guidance of Framework Development. Software –Concepts and Tools*. Editora Springer-Verlager, 1995

SCHMIDT, Hans Albrecht. *Systematic Framework Design by Generalization*.Communications of the ACM, Vol. 40, No.10, Outubro 1997

BARBOSA, Álvaro C. Pereira; LUCENA, Carlos José P. *Integração de Frameworks de Software*. Monografia em Ciência da Computação, PUC-RJ,Departamento de Informática, 2000.