Composição de Fluxo de Controle em Frameworks Java

Bruno Martins Moutinho
Mestrando em Ciência da Computação
Instituto de Matemática e Estatística - Universidade de São Paulo (IME-USP)
bruno@ime.usp.br

Orientadora: *Ana Cristina Vieira de Melo*
Professora Doutora do Departamento de Ciência da Computação
Instituto de Matemática e Estatística - Universidade de São Paulo (IME-USP)
acvm@ime.usp.br

RESUMO

Um dos principais motivos para se utilizar *frameworks* é a reutilização de software, alcançando com isso reutilização de código, *design* e fluxo de controle. Hoje, o desenvolvimento de aplicação baseada em *frameworks* está mudando de baseada em um único *framework* para baseada em múltiplos *frameworks*. Contudo, a maioria dos *frameworks* não foi projetado para ser compostos com outros *frameworks* ou componentes (biblioteca de classes, componentes legados ou *design patterns*), mas para ser reutilizado individualmente. Então, quando compomos vários *frameworks*, surgem problemas tais como: composição de fluxo de controle de *framework*, composição com sistemas legados, *frameworks gap*, sobreposição de entidades e composição de funcionalidade de entidade.

O objetivo principal deste trabalho é estudar técnicas de composição de *frameworks* e integração dos fluxos de controle embutidos para a construção de um terceiro *framework*, e utilizar exemplos das técnicas em *frameworks* construídos com a linguagem Java.

ABSTRACT

One of the main reasons to use frameworks is software reuse, attaining with this reuse of code, design, and flow of control. Today, the framework-based application development is changing from based on single framework to based on multiple frameworks. However, the majority of frameworks was not designed to be composed with others frameworks or components (class library, legacy components or design patterns), but to be reuse individually. Then, when frameworks have to be composed, several problems appear while doing this, related to: composition of framework control, composition with legacy components, framework gap, overlap of framework entities and composition of entity functionality.

The main objective of this work is to study frameworks-composition techniques and integration of control flow embedded to the construction of a third framework, and to use examples of the techniques in frameworks designed with Java language.

Palavras Chaves: *Frameworks* OO; Reutilização de *Frameworks* OO; Composição de *Frameworks* OO; Composição de Fluxo de Controle e *Frameworks* Java.

1 – Introdução

"Um *framework* é um *design* e uma implementação genérica reutilizáveis do comportamento e estruturas dinâmicas de objetos de uma solução geral para um ou mais problemas em um certo domínio" [BEN97]. Os *frameworks* são, basicamente, uma coleção de classes abstratas e concretas e a relação existente entre estas. Dessa forma, ele pode ser visto como uma arquitetura para vários subsistemas. *Frameworks* são importantes porque eles reutilizam análise, *design*, código e fluxo de controle [MAT97].

Atualmente, experiências significativas de reutilização de projeto envolvem a utilização de *frameworks*, ao invés de componentes isolados de software. O desenvolvimento de aplicações baseadas em *frameworks*, depois de um período de aprendizado, diminui o custo e o esforço de desenvolvimento, por isso *frameworks* têm valor comercial [SPA96]. As aplicações estão cada vez mais baseada em múltiplos *frameworks*. Porém, *frameworks* foram projetados para ser estendidos, não compostos com outros *frameworks* [MAT97]. Desta forma, quando dois *frameworks* são compostos surgem problemas, tal como o tratado nesse trabalho composição de fluxo de controle em *frameworks*.

1.1 – Fluxo de Controle em Frameworks (Inversão de Controle)

Diferente de bibliotecas de classes, alguns *frameworks* incorporam um modelo de controle (fluxo de execução). Um *framework* gerencia o fluxo de execução chamando o código da aplicação quando apropriado, bastando para isso, implementar o código específico da aplicação que será chamado pelo *framework* (*callbacks*).

Em lugar da aplicação do usuário, o *framework* normalmente executa o papel de "programa principal" e coordena as atividades da aplicação. A inversão de controle dá ao *framework* o poder de servir como "esqueleto" extensível. Os métodos fornecidos pelo usuário adaptam os algoritmos genéricos definidos no *framework* para uma aplicação particular.

O fluxo de controle em um *framework* é a ordem em que métodos são chamados e eventos são entregues. O fluxo de controle no *framework* está freqüentemente distribuído sobre um grande número de objetos de classes diferentes. Quando um *framework* tem o fluxo de controle da aplicação ele controla todos os eventos daquela aplicação e decide que métodos chamar, iniciar e que tarefas específicas deve executar.

Existem dois tipos de *frameworks* [SPA96]: "que chamam" e "chamados". *Frameworks* "que chamam" são entidades ativas em uma aplicação, controlando e invocando outras partes. Esses *frameworks* têm um modelo de controle bem definido. Por outro lado, *frameworks* "chamados" são entidades passivas que são chamados pela aplicação quando apropriado, esse tipo de *framework* não contém o fluxo de controle, ele é parecido com uma biblioteca de classes.

2 – Composição de Frameworks OO

O desenvolvimento de aplicações é, cada vez mais, baseado em múltiplos *frameworks* que precisam ser compostos com outros componentes (*frameworks*, bibliotecas de classes, *design patterns* e/ou componentes legados). Composição de *frameworks* pode ser utilizada de duas maneiras: (i) compor *frameworks* para criar um novo *framework* com as características dos primeiros e opcionalmente algumas características adicionais, e (ii) criar uma aplicação derivada de vários *frameworks*, nesse caso o produto resultante é uma aplicação que tem como base vários *frameworks*.

Os problemas primários em composição de *frameworks* são: composição de fluxo de controle, composição com sistemas legados, *frameworks gap*, sobreposição de composição de entidades e composição de funcionalidade de entidade. As causas desses problemas são:

comportamento coesivo, cobertura do domínio, intenção do *design* e falta de acesso ao código fonte [MAT97].

2.1 – Composição de Fluxo de Controle em Frameworks

Composição de *frameworks* requer também compor fluxos de controle. Para realizar tal tarefa, temos que primeiro identificar os locais onde os fluxos de controle dos diferentes *frameworks* estão em conflito, para em seguida identificar onde fluxos de controle precisam ser adaptados. O problema de composição de fluxo de controle surge quando se deseja compor *frameworks* "que chamam", uma vez que, ambos esperam ser a entidade gerenciadora do fluxo de controle da aplicação. Outro problema vem do fato que o fluxo de controle freqüentemente não está localizado em uma única entidade, dificultando a composição.

As causas primárias do problema de composição de fluxo de controle em *frameworks* são [MAT97]: (i) intenção de *design*, *frameworks* são intencionalmente projetados para ser reutilizados, geralmente por adaptação, mas não por composição; (ii) comportamento coesivo, indica o comportamento das classes do *framework* para atualizar classes de outros *frameworks*; e (iii) falta de acesso ao código fonte, caso não esteja disponível, a única maneira de se alcançar o comportamento adicional exigido pelo *framework* é através de uma envoltura (*wrapping*) encapsulada ao *framework*, porém, pode causar problemas de desempenho devido ao código adicional.

Aspectos importantes do modelo de controle para a composição são: os recursos que o *framework* gerencia, se ele usa simples ou múltiplos fluxos de execução e a forma que o fluxo de controle *framework* foi implementado (baseado em eventos ou seqüencial).

3 – Trabalhos Similares

Os trabalhos sobre composição de *frameworks* são normalmente para a construção de uma aplicação baseada em mais de um *framework* e poucos trabalhos existentes são sobre Java, a maioria aborda C++ ou independe da linguagem, exemplos são:

- Em [PYA96] é apresentado um *framework* OO desenvolvido em C++, a composição de fluxo de controle foi resolvida utilizando um *design pattern* (REACTOR) que fornece um mecanismo que integra os eventos desmultiplexando e despachando eventos de múltiplos *frameworks*.
- Em [MAT97] os autores apresentam o problema de fluxo de controle e indicam algumas soluções gerais, por exemplo: concorrência (dá a cada *framework* seu próprio fluxo de controle), envoltura (encapsular cada *framework* por um "*wrapper*" interceptando todas as mensagens enviadas para e pelo *framework*) e rescrever o novo fluxo de controle.
- O artigo [CRN00] mostra como fazer a composição de *frameworks* especificados formalmente em lógica computacional. *Frameworks* são divididos em aspectos estáticos (lógica de primeira ordem) e dinâmicos (transição de estados e estrutura de eventos). No artigo os autores mostram como compor estrutura de eventos que são parecidas com o conceito de fluxo de controle.

4 – Abordagem

O contexto do problema é a situação onde dois *frameworks* "que chamam" precisam ser compostos e ambos assumem gerenciar do fluxo de controle (laço principal da aplicação). Quando dois *frameworks* são compostos, os seus fluxos de controle devem ser compostos de alguma maneira. O objetivo principal deste trabalho é estudar técnicas de composição de

frameworks e integração dos fluxos de controle embutidos para a construção de um terceiro *framework*, e utilizar exemplos das técnicas em *frameworks* construídos com a linguagem Java.

A composição entre frameworks é realizada de duas formas: através de troca de mensagens ou através de eventos. Para cada técnica serão estudados os problemas de composição, se o problema pode ou não ser detectado (estaticamente ou dinamicamente) e uma possível solução para o problema (caso exista).

A composição através da troca de mensagens pode ser realizada de quatro formas: sem troca de mensagens, com troca de mensagens subdividida em seqüencial (um *framework* executa primeiro e depois inicia o fluxo de controle do outro *framework*), unidirecional (somente um *framework* envia uma ou mais mensagens) e bidirecional (ambos os *frameworks* enviam mensagens).

Os tipos de composição por eventos são: sem eventos compartilhados (domínios disjuntos) e com eventos compartilhados subdivididos em: eventos internos (somente eventos gerados pelos *frameworks* são iguais), eventos externos (somente eventos originados pela aplicação são iguais) e eventos internos e externos (uma combinação dos dois).

A técnica desenvolvida deve ser capaz de gerenciar os fluxos de controles dos dois *frameworks* primários e evitar interferências indevidas e erros durante a execução. O *framework* deve ser composto com outros *frameworks* sem afetar a sua interface.

5 – Resultados Esperados

Os resultados esperados do trabalho são: (i) uma técnica de composição de fluxo de controle de *frameworks* Java; e (ii) uma ferramenta que será desenvolvida baseada nas técnicas desenvolvidas para automatizar parte da composição de fluxos de controle em *frameworks* Java. A ferramenta será desenvolvida utilizando Java.

5.1 – Estágio Atual

O estágio atual do trabalho está no estudo e desenvolvimento de técnicas de composição de controles de *frameworks* Java, baseada nas técnicas de composição de *frameworks* estudadas (encontradas na literatura) e do mecanismo de controle de Java. As próximas etapas são: implementar as técnicas desenvolvidas na fase anterior aplicadas em *frameworks* Java; comparar a técnica desenvolvida com outras abordagens para resolver o problema de composição de fluxo de controle e implementação da ferramenta.

6 – Bibliografia

[BEN97] BENGTSSON, PerOlof, 1997. *Object Oriented Frameworks: Development issues*. Ericsson Software Technology AB Frameworks, (TF-97:028 rev A), 25 pp.

[CRN00] CRNKOVIC, Ivica; FILIPE, Juliana; LARSSON, Magnus e LAU, Kung-Kiu. *Object-Oriented Design Frameworks: Formal Specification and Some Implementation Issues*.

[MAT97] MATTSSON, Michael; BOSCH, Jan 1997. *Framework Composition: Problems, Causes and Solutions*. Proceedings TOOLS USA'97.

[PYA96] PYARALI, Irfan; HARRISON, Timothy H.; SCHMIDT, D. C. 1996. *Design and Performance of an Object-Oriented Framework for High-Speed Electronic Medical Imaging.* Computing Systems Journal, USENIX, Vol 9, No 4.

[SPA96] SPARKS, S.; BENNER, K.; FARIS, C. *Managing Object-Oriented Framework Reuse*, IEEE Computer, 53-62, September, 1996.