NextFlow: Um Framework para Mapeamento de Processos de Negócio e Objetos

Rógel Garcia de Oliveira, Marco Túlio Valente

[1]Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

{rogelgarcia,mtov}@dcc.ufmg.br

Abstract. *Business Process Management Systems (BPMS) and Information Systems usually follow different programming and architectural paradigms. As a consequence, developers face important challenges when integrating both systems. In this tool paper we describe NextFlow, an Object-Business Process mapping framework designed to ease and to foster the use of BPMS by conventional information systems. NextFlow's design is inspired by the object-relational mapping (ORM) frameworks that are widely used to integrate information systems with relational database management systems.*

Resumo. *Sistemas Gerenciadores de Processos de Negócio (BPMS) e Sistemas de Informação geralmente seguem paradigmas diferentes de programação e arquitetura. Como consequência, desenvolvedores enfrentam desafios importantes quando tem que integrá-los. Nesse artigo nós descrevemos o NextFlow, um framework de mapeamento Objetos-Processos de Negócio criado para facilitar e incentivar a utilização de BPMSs por sistemas de informação convencionais. O projeto do NextFlow foi inspirado pelos frameworks objeto-relacionais (ORM) que são amplamente utilizados para integrar sistemas de informação com sistemas gerenciadores de bancos de dados relacionais.*

1. Introdução

BPMS (*Business Process Management Systems*) são sistemas que permitem a definição e execução de processos de negócio [Aalst et al. 2003]. Utilizando esse tipo de software é possível aliviar parte dos requisitos que deveriam ser implementados em sistemas de informação (SI). BPMSs geralmente proveem linguagens gráficas, que facilitam a participação de analistas de negócio na criação dos fluxos e ainda possuem ferramentas para gerenciamento e testes de processos. No entanto, para aproveitar as funcionalidades oferecidas pelos BPMSs é necessário que exista uma camada de integração com o sistema de informação. Porém, essa integração apresenta pelo menos os seguintes desafios:

- Diferença de paradigma entre BPMSs e SIs [Cardoso et al. 2004]. Mais especificamente, arquiteturas de BPMS não foram desenhadas para integração com sistemas de informação. Ao invés disso, elas tentam incorporar funcionalidades como interface com o usuário e persistência de dados presentes no sistema de informação [Manolescu 2001, Cardoso et al. 2004, Youakim 2008].
- Falta de padronização. BPMSs se valem de uma gama variada de conceitos e APIs que dificultam a integração com sistemas de informação. Como resultado, a integração com um BPMS é única, e não pode ser reaproveitada em outros sistemas [Aalst et al. 2003, Wohed et al. 2009].

Com o objetivo de tratar os problemas de integração entre BPMSs e sistemas de informação, apresentamos nesse artigo a ferramenta NextFlow. Basicamente, NextFlow é um *framework* que permite o mapeamento de elementos de processos de negócio em elementos orientados a objetos. Com isso, é possível acessar processos através de estruturas OO tradicionais.

2. Solução Proposta

O *framework* NextFlow permite a associação entre elementos típicos de sistemas orientados a objetos e processos de negócio. Em tempo de execução, o NextFlow traduz a semântica de um domínio em outro utilizando um conjunto de regras de mapeamento pré-estabelecidas. A Figura 1 resume o processo de utilização do NextFlow. Com base no processo de negócio, o desenvolvedor implementa no sistema de informação classes e métodos que refletem sua estrutura (passo 1). Em tempo de execução, o sistema de informação utiliza os elementos mapeados (passo 2). O NextFlow intercepta as chamadas de métodos (passo 3), que são traduzidas em tarefas a ser executadas no BPMS (passo 4).

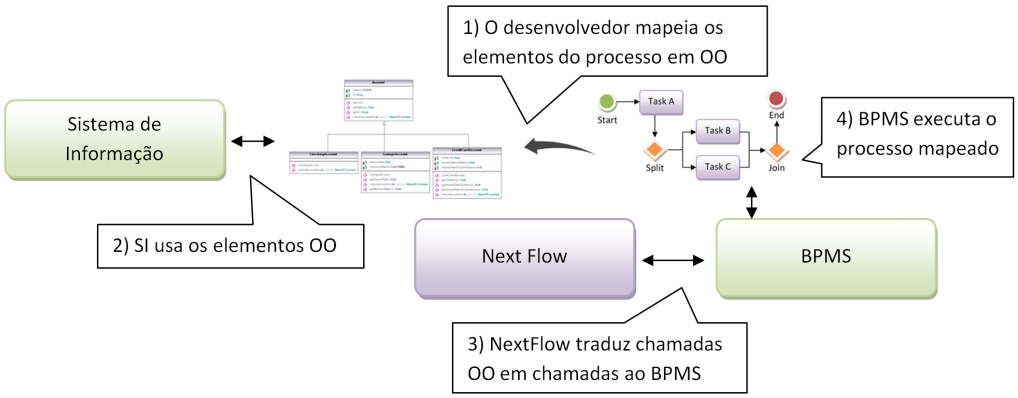

Figura 1. Uso do NextFlow para integração entre SIs e BPMSs

2.1. Modelos de Processos de Negócio

O modelo de processo de negócio utilizado no NextFlow possui apenas elementos genéricos que conseguem representar uma grande variedade de elementos específicos fornecidos pelos BPMSs. A Figura 2 mostra os elementos do modelo utilizado pelo NextFlow. Esse modelo possui dois elementos para representar o início e o final do fluxo (*start*, *end*), elementos para representar paralelismo e sincronização (*split*, *join*), e elementos para representar tarefas (*tasks*).

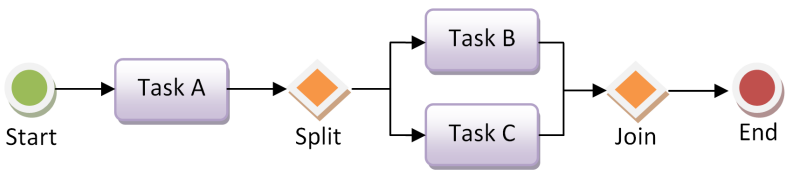

Figura 2. Modelo de processo de negócio

2.2. Regras de Mapeamento

O NextFlow permite o mapeamento dos seguintes artefatos OO para um dado processo de negócio: 1) interfaces de processo, que permitem ao sistema de informação executar tarefas do processo de negócio; 2) classes de dados, que permitem o intercâmbio de informações entre sistemas de informação e BPMSs; e 3) classes de *callback*, que permitem que o BPMS execute operações no sistema de informação.

Interface de Processo. A Figura 3 mostra um exemplo de mapeamento de um processo em uma interface OO. Cada método da interface representa uma tarefa do processo. De posse de um objeto de uma classe que implementa essa interface, basta chamar o método `approveTransaction()` para que a respectiva tarefa seja executada no BPMS.

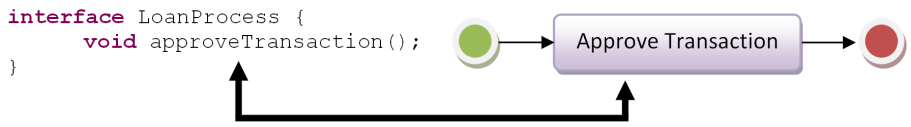

Figura 3. Mapeamento entre processo de negócio e interface de processo

Classe de dados. Um processo de negócio possui um conjunto de informações que são representadas como variáveis globais. Para acessar essas informações, é necessário criar uma classe de dados. Cada atributo dessa classe representa uma variável global do processo. A Listagem 1 representa uma possível classe de dados para o processo mencionado anteriormente.

```
1 class LoanData {
2     String clientID;
3     String getClientID(){return clientID;}
4     void setClientID(String id){clientID = id;}
5 }
```

Listagem 1. Classe representando os dados do processo

Um método de acesso (com prefixo `get`) pode ser adicionado a uma interface de processo para fornecer acesso a objetos de uma classe de dados, como mostrado na Listagem 2 (linha 2).

```
1 interface LoanProcess {
2     LoanData getLoanData();
3     void approveTransaction();
4 }
```

Listagem 2. Método `getLoanData` para acessar dados do processo

Uma chamada `loanProcess.getLoanData().getClientID()` retornará o valor armazenado no processo gerenciado pelo BPMS. O NextFlow coordena toda a comunicação com o BPMS. Da mesma forma, uma chamada ao método `setClientID` atualiza o valor da variável no BPMS.

Callbacks. *Callbacks* são classes que complementam a funcionalidade padrão de uma tarefa. A Listagem 3 mostra uma classe de *callback* para o processo de exemplo. Sempre que a tarefa *approve transaction* for disparada no BPMS, o método `approveTransaction` dessa classe é executado (linha 3). Uma classe de *callback* pode ainda acessar objetos de classes de dados através de atributos (linha 2).

```
1 class LoanProcessCallback {
2     LoanData data;
3     void approveTransaction() {
4         // código que executa a transação
5     }
6 }
```

Listagem 3. Classe de callback

3. Arquitetura do NextFlow

Os objetivos do NextFlow podem ser divididos em duas partes. Primeiro, garantir independência de BPMSs. Segundo, disponibilizar um mapeamento entre elementos do processo (de mais baixo nível) em elementos orientados a objetos (de mais alto nível). Esses dois objetivos são refletidos na arquitetura do NextFlow que possui duas camadas: *Workflow Connectivity* e *Object-Workflow Mapping*. A primeira camada oferece uma interface genérica capaz de representar processos de negócio. A segunda camada utiliza essa interface genérica e permite que elementos OO sejam associados a processos de negócio. A Figura 4 mostra como a arquitetura do NextFlow é organizada.

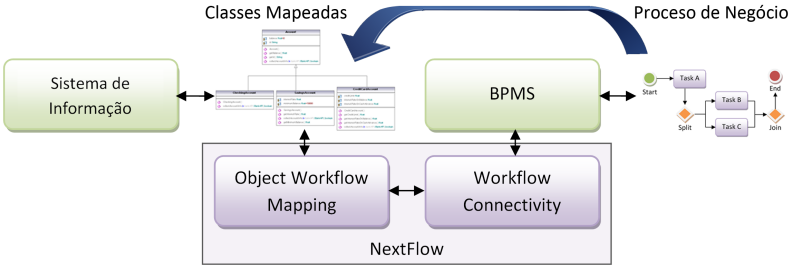

Figura 4. Arquitetura do NextFlow

Em analogia com sistemas gerenciadores de banco de dados, a solução do NextFlow é semelhante aos sistemas JDBC e Hibernate, bastante populares em programas Java. A camada *Workflow Connectivity* fornece independência da implementação do BPMS (sendo, portanto, equivalente a JDBC) e a camada *Object-Worflow Mapping* permite o mapeamento OO (sendo, portanto, equivalente a um *framework* para mapeamento entre objetos e banco de dados relacionais, como o Hibernate). Essas duas camadas são brevemente descritas nas próximas subseções.

3.1. Workflow Connectivity

A camada *Workflow Connectivity* (WFC) provê interfaces genéricas para representar os elementos de um processo de negócio. Portanto, elas não estão associadas a nenhum BPMS específico. A Listagem 4 mostra um exemplo de utilização dessa API onde uma instância de processo é criada (linha 2) e sua atividade *task1* executada (linhas 3-4).

```
1 Session s = WorkflowManager.getSession("jwfc:bpmsX:resources");
2 ProcessInstance pi = s.createProcessInstance("processA");
3 ActivityInstance ai = pi.getActivityByName("task1");
4 ai.complete();
```

Listagem 4. Executando uma tarefa com a API WFC

A implementação das interfaces WFC é fornecida por ***drivers***. Cada *driver* implementa as interfaces do WFC para um BPMS específico. O método `getSession` procura pelos *drivers* disponíveis na aplicação e instancia uma sessão específica para o BPMS solicitado. Esse sistema de *drivers* também é utilizado no JDBC e segue o padrão de projeto Adapter [Gamma et al. 1995].

3.2. Object-Workflow Mapping

A segunda camada do NextFlow, chamada *Object-Workflow Mapping* (OWM), cria implementações em tempo de execução para interfaces mapeadas para processos de negócio. Uma implementação para o processo da Figura 3 é mostrada na Listagem 5.

```
1  class LoanProcessImpl implements LoanProcess {
2    //ProcessInstance is a type provided by the WFC layer
3    ProcessInstance pi;
4    void setProcessInstance(ProcessInstance pi){
5      this.pi = pi;
6    }
7    //methods defined in the LoanProcess interface
8    void approveTransaction(){
9      pi.getActivityByName("approveTransaction").complete();
10   }
11   void taskB(){
12     pi.getActivityByName("taskB").complete();
13   }
14 }
```

Listagem 5. Exemplo de implementação de interface

Para conseguir uma instancia de processos, o OWM provê uma API. Um exemplo típico de utilização dessa API é mostrado na Listagem 6. Primeiro, é necessário fazer a configuração do *framework* (linhas 2-3), o que resulta na criação de um objeto `WorkflowObjectFactory`. Então, por meio dessa fábrica, é possível instanciar processos de negócio que são representados pela interface mapeada (linha 5).

```
1 //configuração
2 Configuration config = new Configuration("jwfc:bpmsX:resources");
3 WorkflowObjectFactory factory = config.createFactory();
4 //acesso ao processo
5 LoanProcess loanProcess = factory.start(LoanProcess.class);
6 loanProcess.approveTransaction();
```

Listagem 6. Utilização da API da camada OWM

Internamente, o método `approveTransaction` (linha 6) acessa o WFC que por sua vez, por meio de um *driver*, delega a operação ao BPMS.

4. Trabalhos Relacionados

Com o objetivo de diminuir a distância entre sistemas de informação e BPMSs outros trabalhos também propõem a utilização de componentes orientados a objetos. Dois exemplos são os sistemas MicroWorkflow [Manolescu 2001] e WebWorkFlow [Hemel et al. 2008]. O MicroWorflow enfatiza a utilização de componentes OO, que podem ser associados para criar um sistema de processos de negócio. Apesar de prover uma arquitetura mais amigável para sistemas de informação, essa solução não oferece vantagens típicas de um BPMS, como linguagens gráficas e um *engine* pronto para

utilização. O WebWorkFlow também propõe a utilização de linguagens OO para criação de processos, porém, a linguagem principal do sistema de informação deve ser alterada. O NextFlow difere dessas duas soluções pois é compatível com linguagens OO tradicionais e BPMSs já estabelecidos e apenas constitui uma camada de integração entre os dois paradigmas.

5. Conclusões

Neste artigo foi apresentado o *framework* NextFlow que permite a integração entre sistemas de informação e BPMSs através de mapeamento dos elementos de processos de negócio em elementos orientados a objetos. O NextFlow oferece uma especificação e uma implementação de regras de associação OO-Processos de Negócio. A abordagem utilizada segue padrões de projetos e características de soluções amplamente utilizadas em sistemas de banco de dados (JDBC e Hibernate). Diferentemente de outras soluções encontradas na literatura, o NextFlow reaproveita linguagens e BPMSs existentes, tratando apenas do ponto de maior dificuldade, que é a integração entre os dois tipos de sistema.

Já foi realizada uma avaliação do NextFlow com duas implementações de um sistema de informação: (a) uma implementação usando o NextFlow; e (b) uma implementação usando a API nativa do sistema jBPM, um dos BPMS mais populares. Os resultados mostraram que o sistema NextFlow oferece uma solução de integração mais simples (redução de 90% do acoplamento com bibliotecas externas), que requer menos esforço de codificação (redução de 75% de código) e que se baseia em abstrações de mais alto nível, tais como interfaces de processo, classes de dados e classes de callback. O NextFlow está disponível publicamente para download em: http://www.nextflow.org. Agradecemos a FAPEMIG e CNPQ pelo apoio ao trabalho.

Referências

Aalst, W., Hofstede, A., and Weske, M. (2003). Business process management: A survey. In *1st International Conference on Business Process Management (BPM)*, pages 1–12.

Cardoso, J., Bostrom, R. P., and Sheth, A. (2004). Workflow management systems and ERP systems: Differences, commonalities, and applications. *Information Technology and Management*, 5(3):319–338.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. Addison-Wesley.

Hemel, Z., Verhaaf, R., and Visser, E. (2008). WebWorkFlow: an object-oriented workflow modeling language for web applications. In *11th Model Driven Engineering Languages and Systems (MODELS)*, pages 113–127.

Manolescu, D. (2001). *Micro-workflow: A Workflow Architecture Supporting Compositional Object-Oriented Software Development*. PhD thesis, University of Illinois.

Wohed, P., Russell, N., Hofstede, A., Andersson, B., and Aalst, W. (2009). Patterns-based evaluation of open source BPM systems: The cases of jBPM, OpenWFE, and Enhydra Shark. *Information and Software Technology*, 51(8):1187–1216.

Youakim, B. (2008). Service-oriented workflow. *Journal of Digital Information Management*, 6(1):118–127.