5
Implementação de operações no HyperDE

A última fase do método SHDM é a Implementação, na qual os modelos produzidos pelas fases anteriores são transformados em uma aplicação executável, em um determinado ambiente de execução.

HyperDE é um ambiente de desenvolvimento rápido que permite o mapeamento de modelos com o objetivo de gerar uma aplicação executável. É um framework MVC (*Model-View-Controller*) composto pelas camadas de modelo navegacional, controle e visão, detalhadas abaixo [Nunes, 2005]:

- A camada de modelo navegacional fornece as primitivas do metamodelo definidas para o SHDM, na qual se pode especificar e manipular os contextos navegacionais, as estruturas de acesso (índices) e seus atributos, as classes navegacionais e seus atributos, os elos, os *landmarks* e os nós – seus atributos e relacionamentos.
- A camada de controle tem como objetivo orquestrar a realização da interação entre usuário e modelo navegacional e usuário e metamodelo da aplicação, por meio da interface *web* do ambiente. Esta camada é composta de diversos controladores, tais como: o controlador de navegação, responsável pelas ações de navegação contextual e construção das estruturas de acesso (índices); e controladores responsáveis pelas ações de manipulação do metamodelo do ambiente e do modelo da aplicação.
- A camada de visão é responsável por prover visões genéricas e fornecer a funcionalidade de gerar visões customizadas. As visões customizadas podem ser associadas a contextos, classes navegacionais e índices.

O Modelo de Operações foi incluído na especificação da camada de modelo do HyperDE.

A seguir é apresentado um diagrama de sequência que ilustra a sequência de colaboração entre os modelos do HyperDE, em um caso de execução de uma operação sobre os dados de uma aplicação.

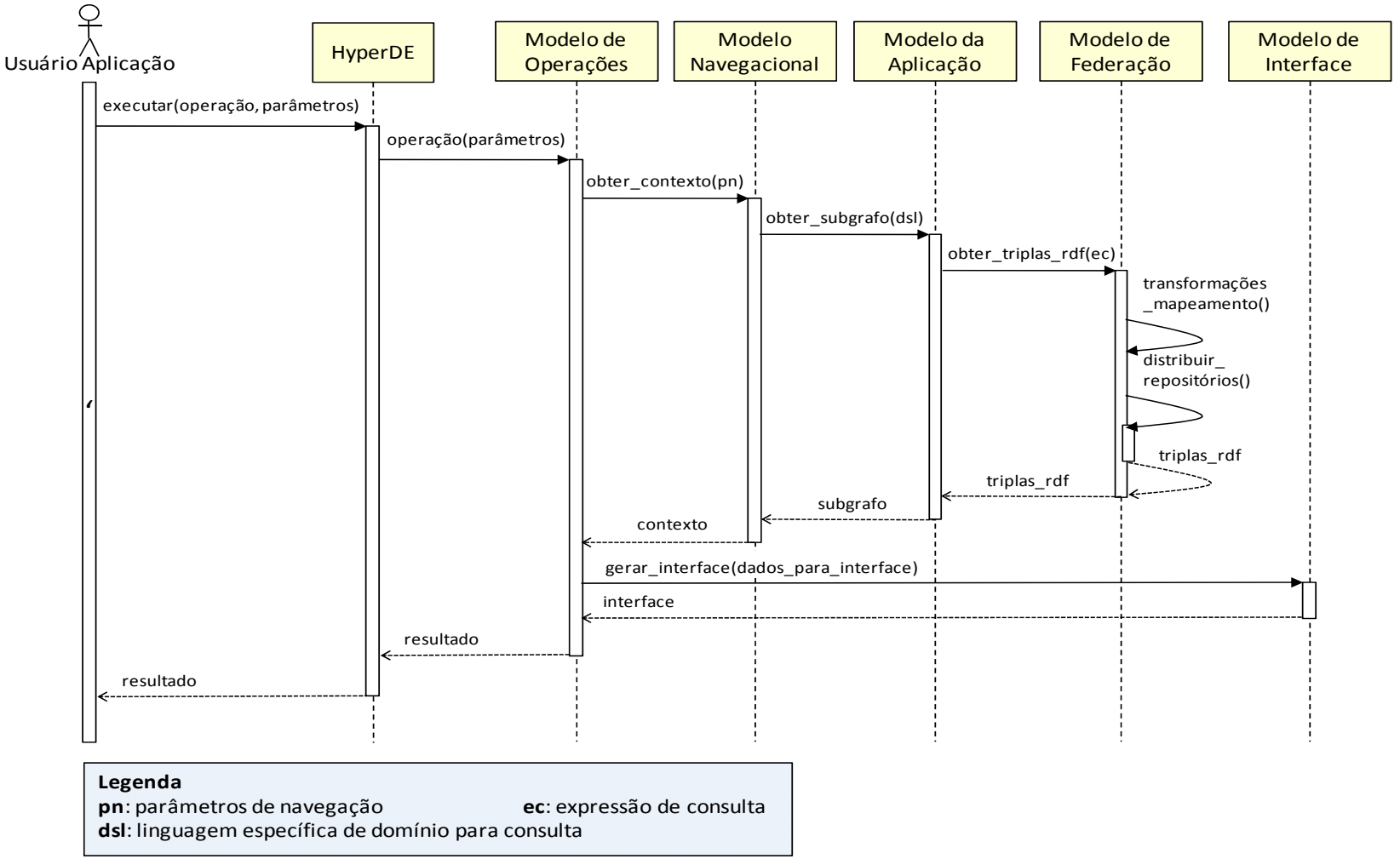

Figura 32 – Colaboração entre os modelos do HyperDE

Na figura 32, um agente externo à aplicação solicita que determinada operação seja executada no HyperDE. O Modelo de Operações é então acionado e decide qual deve ser o comportamento no Modelo Navegacional. As triplas RDF requisitadas são buscadas no banco de dados e retornadas ao Modelo de Operações, que verifica o conteúdo recebido e renderiza a interface de acordo com este resultado. O resultado é apresentado ao agente externo. O Modelo de Federação não será detalhado neste trabalho.

O ambiente HyperDE foi construído utilizando a linguagem de programação dinâmica Ruby, Isso se deve ao fato de que linguagens dinâmicas são mais adequadas para lidar com os modelos de descrição de dados presentes na *web* semântica [Oren et al., 2007]. O RDF é o modelo de descrição de dados utilizado na construção das aplicações geradas pelo HyperDE.

5.1. Classes do Modelo de Operações

A seguir serão apresentadas as classes utilizadas para definição de uma operação em uma aplicação desenvolvida no HyperDE.

5.1.1. Classe Operation – Operações

Instâncias da classe Operation definem operações que expressam a lógica de domínio de uma aplicação. Ela possui relação com as classes OperationParameter, PreCondition e PostCondition e possui como atributos o nome, a linguagem de programação que será utilizada, o tipo e o código da operação.

O atributo linguagem de programação está presente na interface de definição da operação, porém suas funcionalidades não estão implementadas. Isso significa que ainda não é possível escolher dentre várias opções de linguagem de programação aquela que mais se adapta à necessidade do usuário para especificar suas operações. Inicialmente as operações são descritas utilizando a linguagem de programação Ruby.

Uma operação pode ser de dois tipos: interna e externa. Uma operação interna não pode ser acessada por entidades externas à aplicação, sendo invocada por outras operações ou pela interface concreta. O objetivo da operação interna é descrever a regra de negócio da aplicação. Uma operação

externa tem como objetivo servir de canal de comunicação entre as entidades externas e a aplicação e, portanto, são disponibilizadas na interface *web* do ambiente. São acessadas via URL do tipo:

```
http://{servidor_HyperDE}/operation/{nome_da_operacao}?
{query_string}
```

Query_string segue a formatação padrão de descrição de parâmetros enviados em uma URL via http conforme RFC 2616 (http://www.w3.org/Protocols/rfc2616/rfc2616).

A classe Operation possui métodos para persistir os metadados sobre as operações e interpretar estes metadados. Toda operação possui um método denominado “execute” responsável por executar o código da operação baseado nos dados referentes à mesma e aos parâmetros de entrada. Esses parâmetros de entrada são aqueles passados como argumentos para o método “execute”.

Na interface *web* do HyperDE, o tipo da operação é definido por meio de um atributo enumerado, que contém os valores interno e externo, correspondendo às operações internas e externas, respectivamente.

A seguir vemos um trecho de código que exemplifica a definição de uma operação externa que invoca e passa parâmetros para uma operação interna de criação de autor na linguagem de programação Ruby.

```
op = Operation.new
op.name = “create_author_ext”
op.language = “Ruby”
op.type = “external”
op.code = “flash[:message] = ""
flash[:warning] = ""
retorno = ""
if request.post?
  name = params["name"]
  birthday = params["birthday"]
  based_near = params["based_near"]
  phone = params["phone"]
  title = params["title"]
  position = params["position"]
  academic_degree = params["academic_degree"]
  email = params["email"]
  login = params["login"]
  password = params["password"]
  retorno = InternalOperation.create_author_int("name"=>name,
"birthday"=>birthday, "based_near"=>based_near, "phone"=>phone,
"title"=>title, "position"=>position,
"academic_degree"=>academic_degree, "email"=>email,
"login"=>login, "password"=>password)
  if retorno == true
    flash[:message] = "Signup successful"
    redirect_to :controller => :navigation
  elsif retorno.is_a? Hash
    flash[:message] = "Signup unsuccessful. #{ retorno [:msg] if
retorno [:error]}"
```

```
    render :inline => View.find_by_name('CreateAuthor').template
  else
    flash[:message] = "Signup unsuccessful."
    render :inline => View.find_by_name('CreateAuthor').template
  end
else
  render :inline => View.find_by_name('CreateAuthor').template
end”
```

5.1.2. Classe OperationParameter – Parâmetros de Entrada

Instâncias da classe OperationParameter são usadas pelo HyperDE para gerar os parâmetros de entrada da aplicação. A classe OperationParameter possui relação de associação com Operation e NavClass. A relação com NavClass define quais são as classes navegacionais que são passadas como parâmetro para uma determinada operação. OperationParameter pode ser de dois tipos: SimpleParameterIn ou NavClassParameterIn. SimpleParameterIn expressa os parâmetros primitivos da linguagem de programação do Ruby, tais como: String, Integer, Float, Date, Hash e List. NavClassParameterIn representa os parâmetros que são instâncias de classes navegacionais definidas no modelo de domínio.

A seguir vemos um trecho de código que exemplifica um parâmetro de entrada para a operação que cria um novo autor.

```
params = OperationParameter.new
params.name = “params_user”
params.datatype = “Hash”
```

5.1.3. Classes PreCondition e PostCondition – Pré-Condições e Pós-Condições

Instâncias de PreCondition e PostCondition definem as pré e pós-condições de uma operação. Estas classes possuem relação de associação com Operation e têm como atributos o nome e o código das condições. A classe PreCondition possui ainda como atributo o tratamento de falhas (Failure_Handling), que é descrito por um código.

O HyperDE, por ter sido construído utilizando a linguagem Ruby, empregou como técnica para descrição de parâmetros de saída, a estrutura de dados Hash. Essa estrutura de dados foi utilizada porque no Ruby não existem múltiplos retornos.

A seguir vemos um trecho de código que exemplifica uma pré-condição para a operação que cria um novo autor. Esta pré-condição verifica se não existem autores cujo nome já esteja cadastrado na aplicação.

```
pre_condition = PreCondition.new
pre_condition.name = “user_already_exists”
pre_condition.code =
“Author_.find_all_by_name(params_user[“name”]).empty?”
```

5.2. Controlador

O HyperDE possui dois conjuntos de controladores: o controlador de navegação representado pela classe NavigationController, que especifica as ações de navegação e o controlador de retaguarda representado pelas classes derivadas de CrudController, que define as ações de desenvolvimento da aplicação modelada segundo o SHDM.

Um novo controlador que inclui os métodos de CrudController foi definido: OperationController, que implementa as ações para gerenciamento de operações. As ações herdadas de CrudController são: list, para listagem das instâncias; new, fornece os dados para inicialização de uma instância; edit, fornece os dados para atualização de uma instância existente; create, executa a ação de criação de uma nova instância; update, executa a ação de atualização de uma instância existente; delete, executa a ação de exclusão de uma instância existente.

OperationController também inclui os métodos de instância do módulo Operations, ou seja, as operações externas, pois oferece a execução desses métodos a uma entidade externa via URL.

5.3. Modelagem da Interface

Uma nova especificação de interface está sendo definida para permitir que os *widgets* da interface concreta sejam mapeados em operações [Luna, 2010]. Dessa forma, as operações serão referenciadas diretamente na interface.

A seguir é ilustrado um exemplo de uma visão customizada que acessa a operação externa “create_author_ext”, responsável pela chamada da operação criar autor. Esta visão não está adaptada à nova interface, pois esta ainda está sendo especificada em um trabalho paralelo a esse [Luna, 2010].

```
<style>
.notice {
```

```
color: green;
font-weight: bold;
padding: 4px;
}
.error { color: red; font-weight: bold }
</style>
<span class="notice"><%= flash[:message] %></span>
<span class="error"><%= flash[:warning] %></span>
<h1> Create New Author</h1>
<br/>

<% form_tag :url => { :controller => :operation, :action =>
:create_author_ext} do %>
<input name="op" value="create_author_ext" type="hidden">
<table>
  <tr>
    <td>
      Name:
    </td>
    <td>
      <input name="name" type="text" style="width: 200px"/>
    </td>
  </tr>
  <tr>
    <td>
      Birth Date:
    </td>
    <td>
      <input name="birthday" type="datetime" style="width:
200px"/>
    </td>
  </tr>
  <tr>
    <td>
      Address:
    </td>
    <td>
      <input name="based_near" type="text" style="width: 200px"/>
    </td>
  </tr>
  <tr>
    <td>
      Phone:
    </td>
    <td>
      <input name="phone" type="text" style="width: 200px"/>
    </td>
  </tr>
  <tr>
    <td>
      Person title:
    </td>
    <td>
      <input name="title" type="text" style="width: 200px"/>
    </td>
  </tr>
  <tr>
    <td>
      Position:
    </td>
    <td>
```

```
      <input name="position" type="text" style="width: 200px"/>
    </td>
  </tr>
  <tr>
    <td>
      Academic degree:
    </td>
    <td>
      <input name="academic_degree" type="text" style="width:
200px"/>
    </td>
  </tr>
  <tr>
    <td>
      Email:
    </td>
    <td>
      <input name="email" type="text" style="width: 200px"/>
    </td>
  </tr
  <tr>
    <td>
      Login:
    </td>
    <td>
      <input name="login" type="text" style="width: 200px"/>
    </td>
  </tr>
  <tr>
    <td>
      Password:
    </td>
    <td>
      <input name="password" type="password" style="width:
200px"/>
    </td>
  </tr>
</table>
<br/>
<input type="submit" value="Create"/>
<% end %>
```

5.4. Interface *web*

No HyperDE, uma nova aba foi criada para descrever as operações. No modelo antigo, as operações eram descritas dentro das classes navegacionais, acessadas através da aba “Classes” na interface da ferramenta.

Nesta nova aba de operações, o usuário tem acesso ao cadastro de pré-condições, parâmetros de entrada, pós-condições, tipo e valor da operação. Para o cadastro do valor da operação, o usuário define o tipo de linguagem que será utilizado. No momento o ambiente suporta apenas a utilização de código Ruby. A figura abaixo ilustra como é a tela de edição de operações no HyperDE.

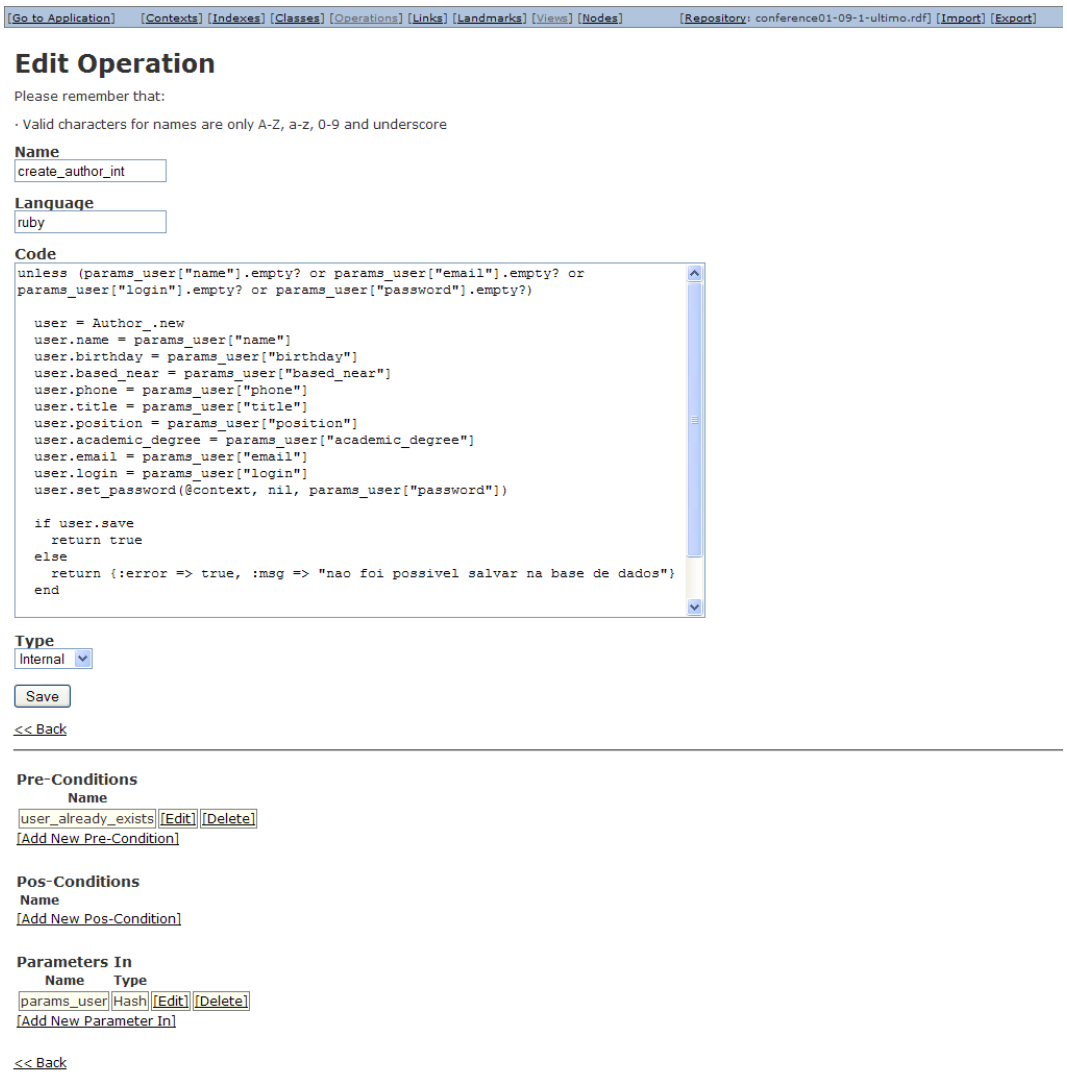

Figura 33 – Tela de cadastro de operações no HyperDE