Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Mapeamento entre geradores de código e uma implementação de referência

Relatório Final - Iniciação Científica

Matheus Sant' Ana Lima
Aluno

Prof. Dr. Daniel Lucrédio
Orientador

São Carlos - SP

2011

1 Resumo dos objetivos e plano inicial

O Desenvolvimento Orientado a Modelos, ou *Model-Driven Development*, é um modelo de desenvolvimento de software que difere do atual paradigma utilizado massivamente na indústria por buscar minimizar a intervenção humana e otimizar a criação de código, seja na criação ou manutenção de sistemas. Para tanto, define-se o conceito conhecido como modelo, que além de permitir a comunicação de ideias, serve de entrada para geradores de código, que produzem como saída aplicações completas. Desta forma, é permitido ao projetista do sistema um nível maior de abstração, sem se preocupar com detalhes pertinentes à implementação.

A proposta desta pesquisa fundamenta-se no uso de geradores baseados em *templates*, juntamente com uma implementação de referência. Os *templates* definem as rotinas e os padrões de código que irão se repetir no programa gerado. Já a implementação de referência serve de exemplo do código a ser gerado e servirá também para alterações, depurações e testes. Depois de validado, o código de referência é migrado para o gerador, ocasionando muitas vezes duplicação entre a implementação de referência e o *template*, dependendo, necessariamente, de intervenção humana para a sincronização dos dois artefatos.

O objetivo deste projeto foi iniciar uma investigação envolvendo técnicas que são um passo inicial para automatizar, ainda que parcialmente, a migração de código. Este estudo se baseia no uso de *templates* para geração de código [2], aliado a uma implementação de referência [5] que serve como exemplo de código a ser gerado [1]. A idéia geral, como pode ser visto na Figura 1, é que uma eventual mudança na implementação de referência possa ser propagada automaticamente para o *template* associado ao trecho modificado. Futuras gerações de código passam a incorporar as mudanças, gerando código consistente com a nova implementação de referência. Com isso, o esforço de manutenção em um processo de MDD é reduzido, uma vez que tanto *templates* quanto a implementação de referência podem evoluir conjuntamente. Também facilita a criação inicial dos *templates*, pois a partir de um *template* mínimo, pode-se introduzir automaticamente modificações incrementais até que o mesmo esteja completo.

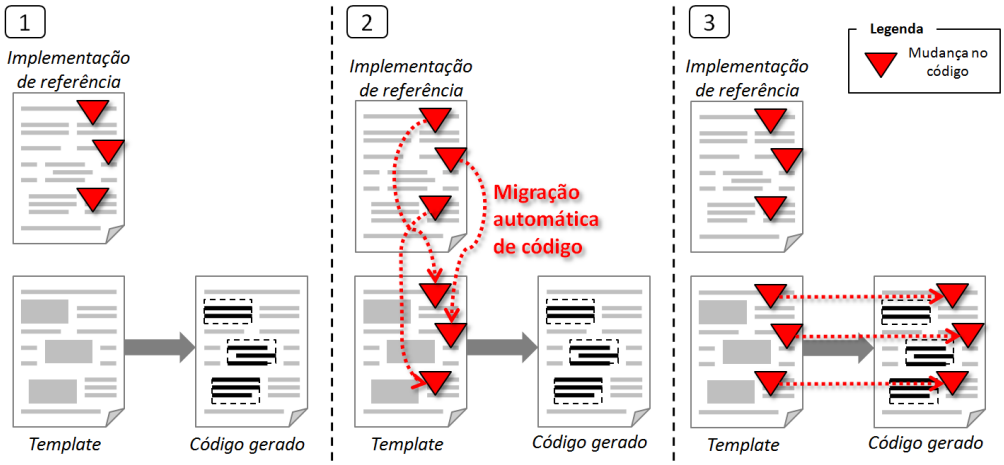

Figura 1: Objetivo do projeto: Migração automática de código

Como ponto de partida em direção a este objetivo maior, este projeto implementou diferentes mapeamentos que foram previamente identificados em um estudo anterior [3], entre um *template* e a implementação de referência. Os mapeamentos foram estabelecidos por meio de técnicas de anotação de código, conforme descrito mais adiante neste relatório.

As seguintes atividades estavam previstas para este projeto:

1. Estudo sobre geradores de código baseados em *templates*: serão estudadas tecnologias de geração de código baseada em *templates*, assim como as ferramentas que serão utilizadas na pesquisa.

2. Identificação dos diferentes tipos de mapeamento entre *templates* e implementação de referência: com base no estudo da tecnologia de geração de código, será feito um levantamento das possibilidades de combinação entre *templates* e uma implementação de referência.

3. Implementação dos mapeamentos identificados: uma vez identificados os mapeamentos, os mesmos serão concretizados por meio de anotações de código. Esta atividade consiste na implementação de mecanismos de geração de código que automaticamente inserem anotações conforme os tipos de mapeamento identificados.

4. Avaliação: consiste na realização de um estudo de caso, para identificar possíveis falhas no mapeamento estabelecido.

2 Resultados e discussão

Esta seção apresenta os resultados encontrados mediante a implementação do mapeamento entre *templates* e implementação de referência. Para possibilitar um melhor entendimento dos resultados e discussões, a próxima subseção apresenta o estudo sobre mapeamentos que serviu como base para a implementação.

2.1 Mapeamento entre templates e implementação de referência

Antes do início deste projeto, foi conduzida uma análise inicial [3] da relação entre *templates* e a implementação de referência, que identificou sete diferentes tipos de mapeamento. Esses mapeamentos identificam as possíveis maneiras com que um *template* produz código, e serviram de ponto de partida este projeto. São eles:

- **Tipo 1 - Cópia simples:** Este tipo de mapeamento é o mais simples. Um pedaço de código em um *template* é copiado para dentro da implementação de referência. Uma relação um-para-um é estabelecida, e o modelo de entrada não é consultado. A Figura 2 mostra este tipo de mapeamento:

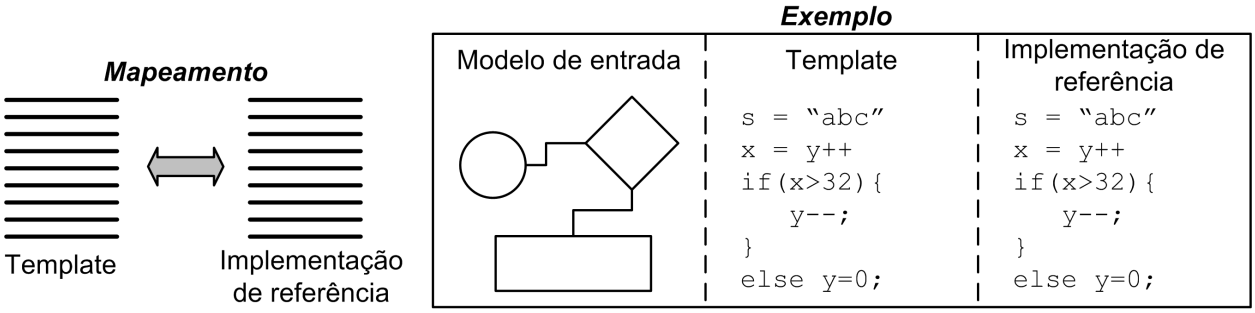

Figura 2: Mapeamento tipo 1.

- **Tipo 2 - Substituição simples:** Uma consulta no modelo de entrada é inserida dentro do template, sendo substituída por algum valor obtido do modelo de entrada quando o código é gerado. É possível rastrear o valor substituído no modelo de entrada, pois se trata de uma simples substituição em um relacionamento um-para-um. A Figura 3 mostra este tipo de mapeamento:

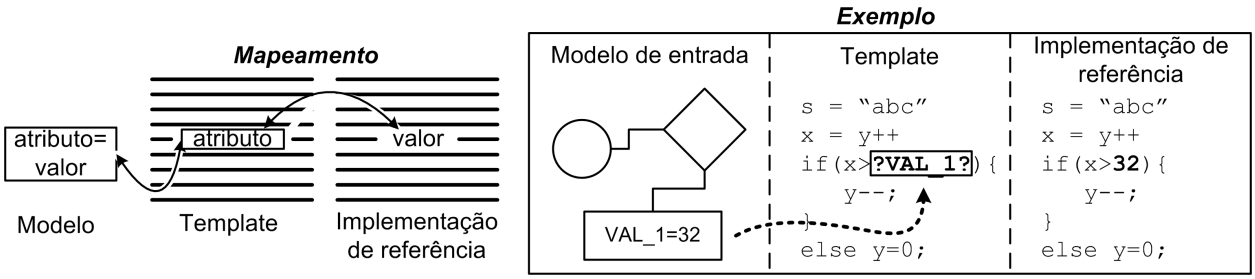

Figura 3: Mapeamento tipo 2.

- **Tipo 3 - Substituição Indireta:** Similar ao tipo 2, mas aqui não é possível rastrear a origem do valor substituído, uma vez que não é explícito no modelo, sendo calculada no *template*. O relacionamento é um-para-um. A Figura 4 mostra este mapeamento:

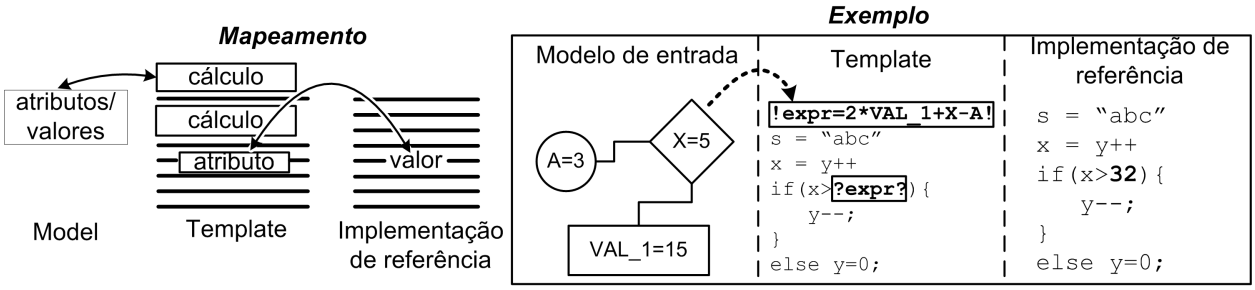

Figura 4: Mapeamento tipo 3.

- **Tipo 4 - repetição:** Um pedaço de código do *template* é repetido zero ou mais vezes na implementação de referência, de acordo com alguns critérios, que podem ser obtidos a partir do modelo de entrada ou não. O relacionamento é um-para-muitos. A Figura 5 mostra esta tipo de mapeamento:

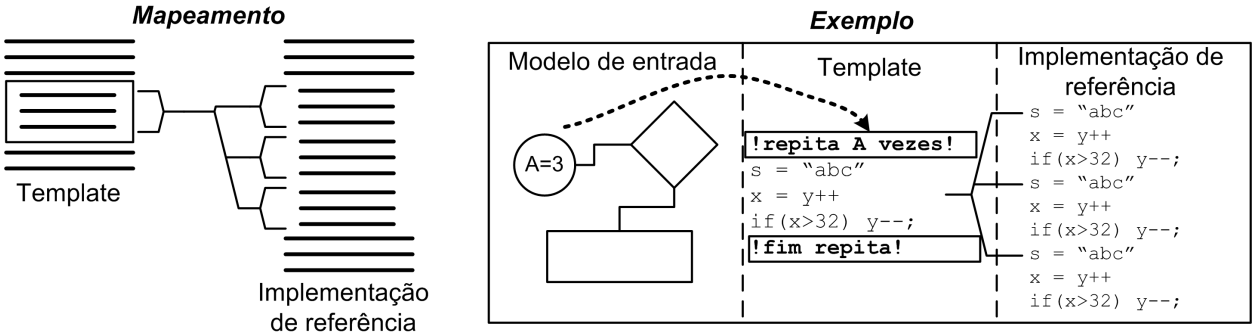

Figura 5: Mapeamento tipo 4.

- **Tipo 5 - condicionais:** Um pedaço da implementação de referência pode ser mapeado para diferentes pedaços de um *template*, baseado em alguma condição

estabelecida no template. A condição pode consultar o modelo de entrada ou não. O relacionamento é muitos-para-um. A Figura 6 mostra este tipo de mapeamento:

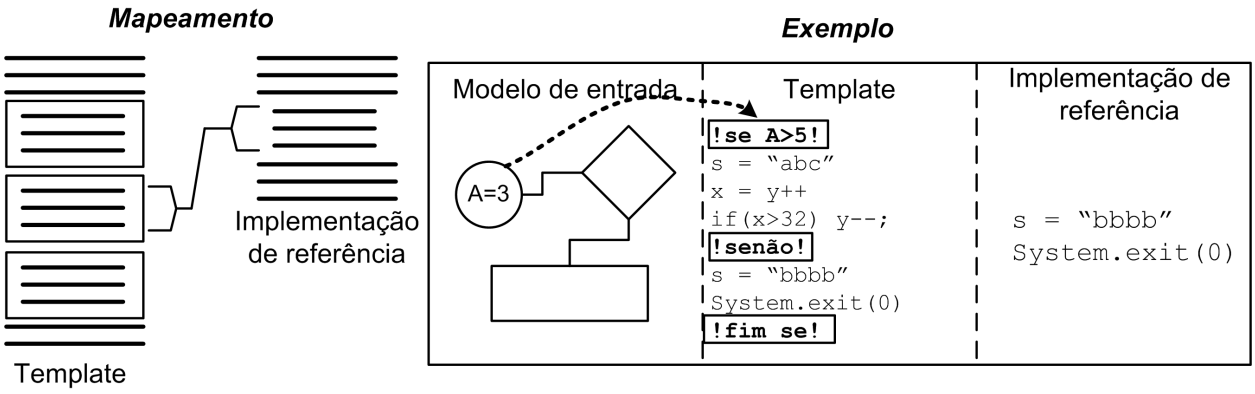

Figura 6: Mapeamento tipo 5.

- **Tipo 6 - inclusão:** Um *template* inclui um outro *template* para gerar uma determinada parte do código. A relação entre os *templates* e a implementação de referência é muitos-para-um. É importante destacar que é sempre possível identificar, para cada linha de código gerado, o *template* exato responsável por sua geração.

- **Tipo 7 - novo arquivo:** Um *template* requisita a criação de um novo arquivo. Após este pedido, o código gerado pelo template é redirecionado para este arquivo recém-criado. O relacionamento entre os *templates* e a implementação de referência é um-para-muitos. Como no tipo 6, é sempre possível identificar, para cada arquivo que contém diferentes códigos gerados, o *template* exato responsável por sua geração. A Figura 7 mostra os mapeamentos tipos 6 e 7.

2.2 Implementação dos mapeamentos

A implementação baseou-se no JET (*Java Emitter Templates*), um popular gerador de códigos baseado em Java. Um *template* JET possuit *tags* que implementam os diferentes mapeamentos descritos na seção anterior.

A implementação consistiu em modificar as *tags* originais do JET, produzindo novas *tags* que, por meio de anotações no código gerado, delimitam e identificam os

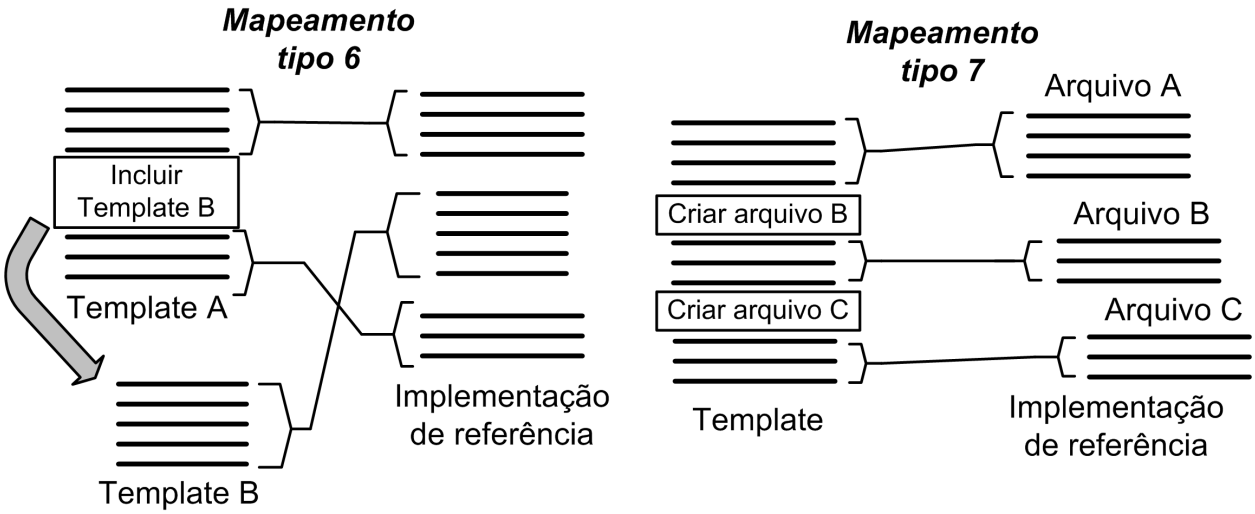

Figura 7: Mapeamentos tipos 6 e 7.

trechos de código envolvidos em cada mapeamento. Cada trecho de código gerado é único e as informações pertinentes à sua criação são importantes para a migração de código.

Cada anotação possui o seguinte formato e conjunto de informações:

```
(// T{i}:{templateName}.jet#{ID})
```

Onde T{i} referencia o tipo de mapeamento; {templateName}.jet referencia o nome do *template* dado como entrada e {ID} é um identificador único para a respectiva *tag*.

2.2.1 Autômato de Pilha e Linguagens Livres de Contexto

A biblioteca JET reconhece padrões no formato XML que marcam um dos sete tipos de mapeamento descritos anteriormente. O *"parser"* ou interpretador, realiza o escrutínio de toda a cadeia de entrada, desta forma, é importante destacar que a gramática que define as novas *tags*, assim como as *tags* JET comuns, são tratadas como Livres de Contexto, uma vez que este tipo de gramática também abrange as Linguagens Regulares, que define algumas *tags*, como a *tag* <m:get>, e é modelada através do conceito de Autômato de Pilha, de fácil implementação.

Para cada padrão reconhecido, um contador é incrementado e seu valor é utilizado para indexar esta *tag* de forma única, no campo {ID}, da anotação gerada. A Figura 8 ilustra a estrutura básica do tipo pilha:

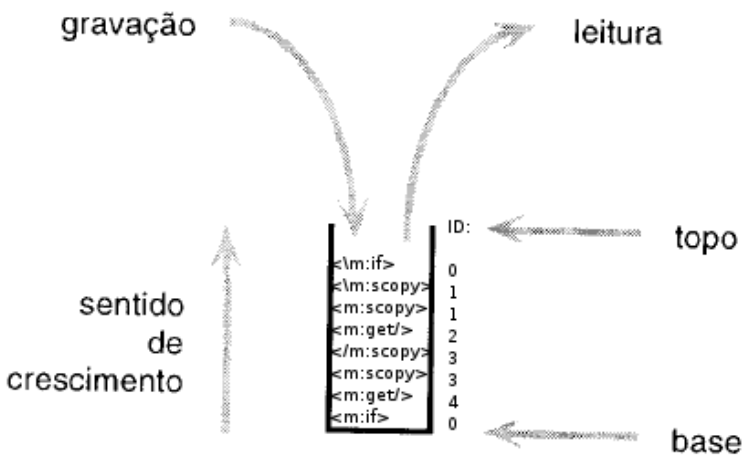

Figura 8: Estrutura do tipo pilha, adaptada de [4].

2.3 Mapeamentos

A seguir são apresentados os códigos gerados a partir de vários *templates* de entrada, utilizando as novas *tags* e as *tags* originais.

2.3.1 Tipo 1 - Cópia simples

Na Figura 9 é ilustrada a geração de código utilizando a nova *tag* para o Mapeamento 1, em comparação às *tags* tradicionais do pacote JET.

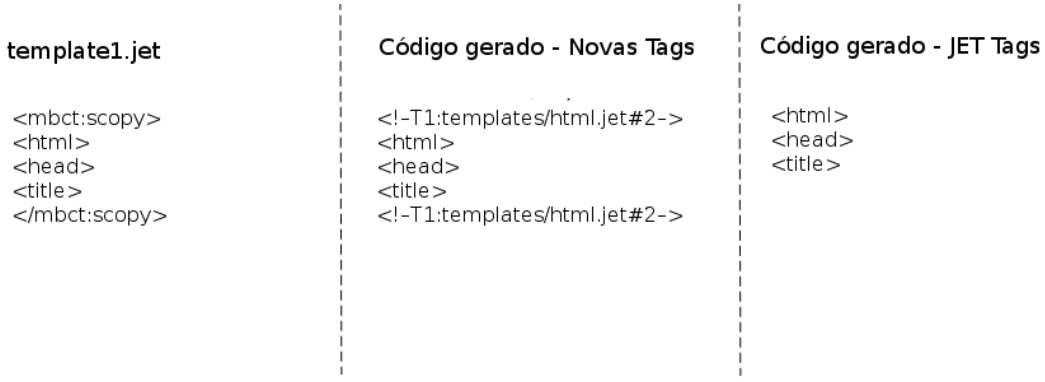

Figura 9: Mapeamento tipo 1 implementado utilizando *tags* personalizadas.

2.3.2 Tipo 2 - Substituição simples

Na Figura 10 é ilustrado a geração de código utilizando a nova *tag* para o Mapeamento 2, em comparação às *tags* tradicionais do pacote JET.

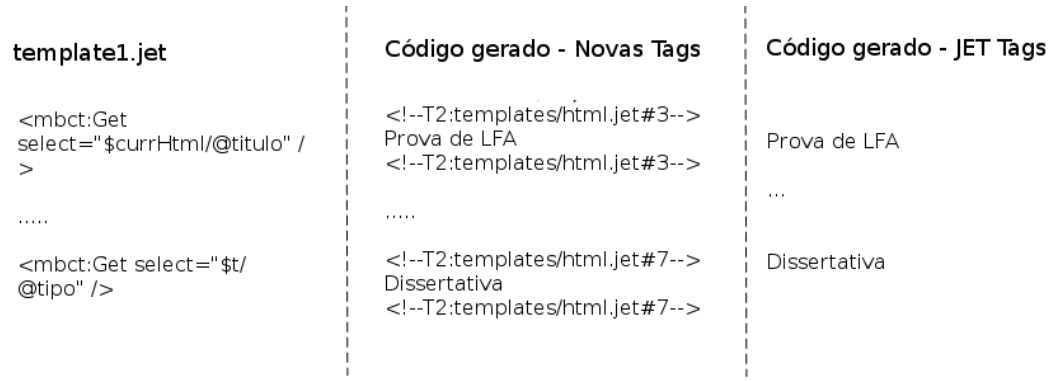

Figura 10: Mapeamento tipo 2 implementado utilizando *tags* personalizadas.

2.3.3 Tipo 3 - Substituição Indireta

Na Figura 11 é ilustrado a geração de código utilizando a nova *tag* para o Mapeamento 3, em comparação às *tags* tradicionais do pacote JET.

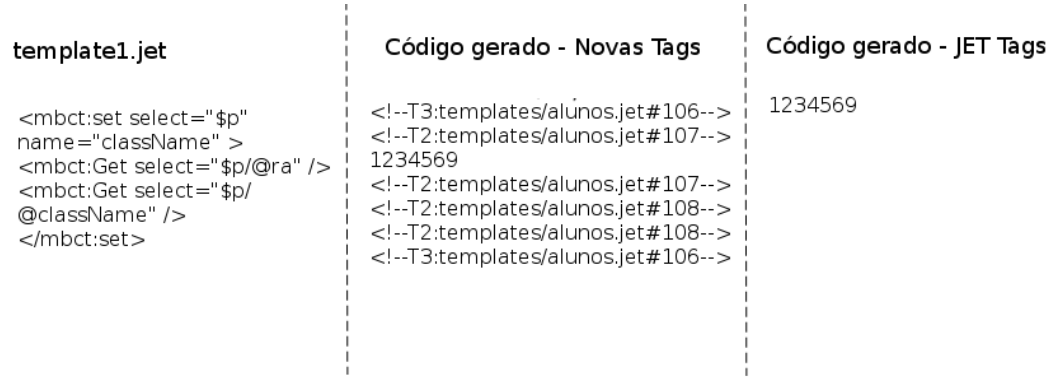

Figura 11: Mapeamento tipo 3 implementado utilizando *tags* personalizadas.

2.3.4 Tipo 4 - repetição

Na Figura 12 é ilustrado a geração de código utilizando a nova *tag* para o Mapeamento 4, em comparação às *tags* tradicionais do pacote JET.

2.3.5 Tipo 5 - condicionais

Na Figura 13 é ilustrado a geração de código utilizando a nova *tag* para o Mapeamento 5, em comparação às *tags* tradicionais do pacote JET.

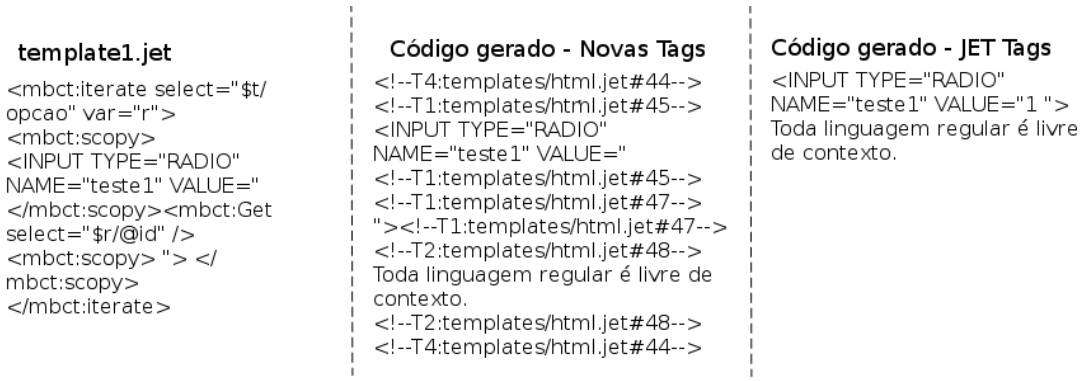

Figura 12: Mapeamento tipo 4 implementado utilizando *tags* personalizadas.

Figura 13: Mapeamento tipo 5 implementado utilizando *tags* personalizadas.

2.3.6 Tipo 6 - inclusão

Na Figura 14 é ilustrado a geração de código utilizando a nova *tag* para o Mapeamento 6, em comparação às *tags* tradicionais do pacote JET.

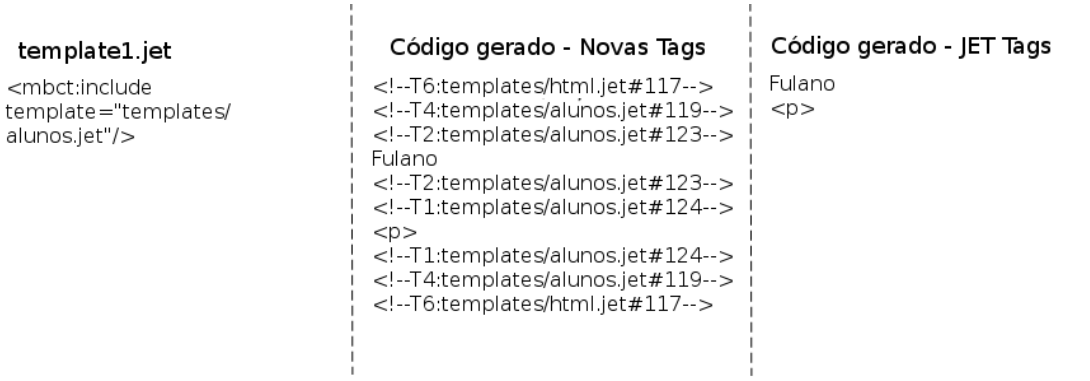

Figura 14: Mapeamento tipo 6 implementado utilizando *tags* personalizadas.

2.3.7 Tipo 7 - novo arquivo

Na Figura 15 é ilustrado a geração de código utilizando a nova *tag* para o Mapeamento 7, em comparação às *tags* tradicionais do pacote JET.

Figura 15: Mapeamento tipo 7 implementado utilizando *tags* personalizadas.

3 Conclusões

Este projeto de pesquisa, tendo como objetivo auxiliar no mapeamento entre *templates* e implementação de referência, procurou dar o passo inicial na redução do esforço no processo de migração de código. O mapeamento consistente é requisito fundamental para a continuidade das investigações, cujo objetivo final é reduzir o esforço adicional de 20% a 25% que é gasto atualmente em todo o esforço de migração de código.

Além dos resultados práticos acima descritos, espera-se também obter importantes publicações, uma vez que este tema é ainda pouco explorado na literatura. Os resultados obtidos e seus pontos fortes e fracos podem interessar à comunidade de maneira a fortalecer o conhecimento nesta área. Como é possível observar na seção 7, o código gerado com as novas *tags* é identico ao código gerado pelas *tags* originais, produzindo desta forma um mapeamento completo do Anexo A, como é reproduzido no Anexo B. Conclui -se, portanto, que que o projeto cumpriu com seus objetivos ao realizar com sucesso tanto uma geração fidedigna para cada uma das novas *tags*, mas também produzir o mapeamento completo do *template* utilizado nesta pesquisa.

Referências

[1] J. Craig Cleaveland. Building application generators. *IEEE Software*, 7(1):25–33, 1988.

[2] Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.

[3] Daniel Lucrédio and Renata Pontin de Mattos Fortes. Mapping code generation templates do a reference implementation - towards automatic code migration. In *I Brazilian Workshop on Model-Driven Development, 2010, Salvador, BA, Brasil*, pages 85–92, 2010.

[4] P. F. Blauth Menezes. *Linguagens Formais e Autômatos - 4a edição*. Inst. de Informática da UFRGS. Editora Sagra Luzzatto, 2005.

[5] Milosz Muszynski. Implementing a domain-specific modeling environment for a family of thick-client gui components. In *The 5th OOPSLA Workshop on Domain-Specific Modeling, San Diego USA*, 2005.

4 Anexo A

Arquivo *html.jet*

```
<mbct:scopy>
<html>
<head>
<title>
</mbct:scopy>
<mbct:Get select='$currHtml/@titulo' />
<mbct:scopy>
</title>
</head>
<body>
</mbct:scopy>

<mbct:iterate select='$currHtml/questao' var='t'>

<mbct:MyIfTag test='$t/@num = '1"'>
<FONT size='3' face='Tahoma' color='blue'>
<mbct:Get select='$t/@tipo' />
<mbct:scopy>
</FONT> <p>
</mbct:scopy>
<mbct:Get select='$t/enunciado' />
<mbct:scopy>
<p> <INPUT TYPE 'TEXT' NAME='nome' SIZE=50 VALUE=''> </p>
</mbct:scopy>
</mbct:MyIfTag>

<mbct:MyIfTag test='$t/@num = '2"'>

<mbct:scopy>
<FONT size='3' face='Tahoma' color='blue'>
</mbct:scopy>

<mbct:Get select='$t/@tipo' />
<mbct:scopy>
</FONT> <p>
</mbct:scopy>
<mbct:iterate select='$t/opcao' var='r'>
<mbct:scopy>
<INPUT TYPE='RADIO' NAME='teste1' VALUE='
</mbct:scopy>
```

```
<mbct:Get select='$r/@id' /><mbct:scopy> '> </mbct:scopy>
<mbct:Get select='$r' />
<mbct:scopy>
<p>
</mbct:scopy>
</mbct:iterate>
</mbct:MyIfTag>

<mbct:MyIfTag test='$t/@num = '3"">
<mbct:scopy>
<FONT size='3' face='Tahoma' color='blue'>
</mbct:scopy>
<mbct:Get select='$t/@tipo' />
<mbct:scopy>
</FONT> <p>
</mbct:scopy>
<mbct:Get select='$t/enunciado' />
<mbct:scopy>
</p><INPUT TYPE='RADIO' NAME='teste2' VALUE='Verdadeiro'>Verdadeiro<p>
<INPUT TYPE='RADIO' NAME='teste2' VALUE='Falso'>Falso<p>
</mbct:scopy>
</mbct:MyIfTag>
<mbct:file template='templates/alunos.jet'
path='$org.eclipse.jet.resource.project.name/gerado/alunos.html' />
<mbct:include template='templates/alunos.jet'/>
</mbct:iterate>
<mbct:scopy>
</body>
</html>
</mbct:scopy>
```

Arquivo *alunos.jet*

```
<mbct:scopy>
Alunos:<p>
</mbct:scopy>

<mbct:iterate select='/questionario/aluno' var='p'>

<mbct:set select='$p' name='className' >
<mbct:Get select='$p/@ra' /><mbct:Get select='$p/@className' /></mbct:set>
<mbct:Get select='$p/nome' />
<mbct:scopy>
<p>
</mbct:scopy>

</mbct:iterate>
```

Arquivo *main.jet*

<mbct:iterate select=’/questionario’ var=’currHtml’ >

<mbct:file template=’templates/html.jet’
path=’{$org.eclipse.jet.resource.project.name}/gerado/teste.html’ />
</mbct:iterate>

5 Anexo B

Arquivo *teste.html*

<!–T1:templates/html.jet#2–>
<html>
<head>
<title>
<!–T1:templates/html.jet#2–>

<!–T2:templates/html.jet#3–>
Prova de LFA
<!–T2:templates/html.jet#3–>

<!–T1:templates/html.jet#4–>
</title>
</head>
<body>
<!–T1:templates/html.jet#4–>

<!–T4:templates/html.jet#5–>

<!–T5:templates/html.jet#6–>
<FONT size=’3’ face=’Tahoma’ color=’blue’>
<!–T2:templates/html.jet#7–>
Dissertativa
<!–T2:templates/html.jet#7–>

<!–T1:templates/html.jet#8–>
</FONT>
<!–T1:templates/html.jet#8–>

<!–T2:templates/html.jet#9–>
Prove que todo automato finito det. é um NFA
<!–T2:templates/html.jet#9–>

<!–T1:templates/html.jet#10–>
 <INPUT TYPE ’TEXT’ NAME=’nome’ SIZE=50 VALUE=”>
<!–T1:templates/html.jet#10–>

<!–T5:templates/html.jet#6–>

<!– T7-Parent:templates/html.jet–>
<!–T7-Child:templates/alunos.jet#11–>

<!–T6:templates/html.jet#25–>

<!–T1:templates/alunos.jet#26–>
Alunos:<p>
<!–T1:templates/alunos.jet#26–>

<!–T4:templates/alunos.jet#27–>

<!–T3:templates/alunos.jet#28–>
<!–T2:templates/alunos.jet#29–>
1234569
<!–T2:templates/alunos.jet#29–>
<!–T2:templates/alunos.jet#30–>

<!–T2:templates/alunos.jet#30–>
<!–T3:templates/alunos.jet#28–>

<!–T2:templates/alunos.jet#31–>
Fulano
<!–T2:templates/alunos.jet#31–>

<!–T1:templates/alunos.jet#32–>
<p> <!–T1:templates/alunos.jet#32–>

<!–T4:templates/alunos.jet#27–>
<!–T4:templates/alunos.jet#33–>

<!–T3:templates/alunos.jet#34–>
<!–T2:templates/alunos.jet#35–>
547862
<!–T2:templates/alunos.jet#35–>
<!–T2:templates/alunos.jet#36–>

<!–T2:templates/alunos.jet#36–>
<!–T3:templates/alunos.jet#34–>

<!–T2:templates/alunos.jet#37–>
Ciclano
<!–T2:templates/alunos.jet#37–>

<!–T1:templates/alunos.jet#38–>
<p> <!–T1:templates/alunos.jet#38–>

<!–T4:templates/alunos.jet#33–>
<!–T6:templates/html.jet#25–>

<!–T4:templates/html.jet#5–>
<!–T4:templates/html.jet#39–>

<!–T5:templates/html.jet#40–>

<!–T1:templates/html.jet#41–>
<FONT size=’3’ face=’Tahoma’ color=’blue’>
<!–T1:templates/html.jet#41–>

<!–T2:templates/html.jet#42–>
MultiplaEscolha
<!–T2:templates/html.jet#42–>

<!–T1:templates/html.jet#43–>
</FONT> <p>
<!–T1:templates/html.jet#43–>

<!–T4:templates/html.jet#44–>
<!–T1:templates/html.jet#45–>
<INPUT TYPE=’RADIO’ NAME=’teste1’ VALUE=’
<!–T1:templates/html.jet#45–>

<!–T2:templates/html.jet#46–>
1
<!–T2:templates/html.jet#46–>
<!–T1:templates/html.jet#47–>
’> <!–T1:templates/html.jet#47–>

<!–T2:templates/html.jet#48–>
Toda linguagem regular é livre de contexto.
<!–T2:templates/html.jet#48–>

<!–T1:templates/html.jet#49–>
<p> <!–T1:templates/html.jet#49–>

<!–T4:templates/html.jet#44–>
<!–T4:templates/html.jet#50–>
<!–T1:templates/html.jet#51–>
<INPUT TYPE=’RADIO’ NAME=’teste1’ VALUE=’
<!–T1:templates/html.jet#51–>

<!–T2:templates/html.jet#52–>
2
<!–T2:templates/html.jet#52–>
<!–T1:templates/html.jet#53–>
’> <!–T1:templates/html.jet#53–>

<!–T2:templates/html.jet#54–>
P = NP.

<!–T2:templates/html.jet#54–>

<!–T1:templates/html.jet#55–>
<p> <!–T1:templates/html.jet#55–>

<!–T4:templates/html.jet#50–>
<!–T4:templates/html.jet#56–>
<!–T1:templates/html.jet#57–>
<INPUT TYPE='RADIO' NAME='teste1' VALUE='
<!–T1:templates/html.jet#57–>

<!–T2:templates/html.jet#58–>
3 <!–T2:templates/html.jet#58–>
<!–T1:templates/html.jet#59–>
'> <!–T1:templates/html.jet#59–>

<!–T2:templates/html.jet#60–>
Não-Determinismo aumenta capacidade de reconhecimento de uma MT.
<!–T2:templates/html.jet#60–>

<!–T1:templates/html.jet#61–>
<p> <!–T1:templates/html.jet#61–>

<!–T4:templates/html.jet#56–>
<!–T4:templates/html.jet#62–>
<!–T1:templates/html.jet#63–>
<INPUT TYPE='RADIO' NAME='teste1' VALUE='
<!–T1:templates/html.jet#63–>

<!–T2:templates/html.jet#64–>
4
<!–T2:templates/html.jet#64–>
<!–T1:templates/html.jet#65–>
'> <!–T1:templates/html.jet#65–>

<!–T2:templates/html.jet#66–>
Nenhuma das anteriores
<!–T2:templates/html.jet#66–>

<!–T1:templates/html.jet#67–>
<p>
<!–T1:templates/html.jet#67–>

<!–T4:templates/html.jet#62–>
<!–T5:templates/html.jet#40–>

<!– T7-Parent:templates/html.jet–>
<!–T7-Child:templates/alunos.jet#68–>

<!–T6:templates/html.jet#82–>

<!–T1:templates/alunos.jet#83–>
Alunos:<p>
<!–T1:templates/alunos.jet#83–>

<!–T4:templates/alunos.jet#84–>

<!–T3:templates/alunos.jet#85–>
<!–T2:templates/alunos.jet#86–>
1234569
<!–T2:templates/alunos.jet#86–>
<!–T2:templates/alunos.jet#87–>

<!–T2:templates/alunos.jet#87–>
<!–T3:templates/alunos.jet#85–>

<!–T2:templates/alunos.jet#88–>
Fulano
<!–T2:templates/alunos.jet#88–>

<!–T1:templates/alunos.jet#89–>
<p> <!–T1:templates/alunos.jet#89–>

<!–T4:templates/alunos.jet#84–>
<!–T4:templates/alunos.jet#90–>

<!–T3:templates/alunos.jet#91–>
<!–T2:templates/alunos.jet#92–>
547862
<!–T2:templates/alunos.jet#92–>
<!–T2:templates/alunos.jet#93–>

<!–T2:templates/alunos.jet#93–>
<!–T3:templates/alunos.jet#91–>

<!–T2:templates/alunos.jet#94–>
Ciclano
<!–T2:templates/alunos.jet#94–>

<!–T1:templates/alunos.jet#95–>
<p> <!–T1:templates/alunos.jet#95–>

<!–T4:templates/alunos.jet#90–>
<!–T6:templates/html.jet#82–>

<!–T4:templates/html.jet#39–>
<!–T4:templates/html.jet#96–>

<!–T5:templates/html.jet#97–>
<!–T1:templates/html.jet#98–>
<FONT size='3' face='Tahoma' color='blue'>
<!–T1:templates/html.jet#98–>

<!–T2:templates/html.jet#99–>
VerdadeiroOuFalso
<!–T2:templates/html.jet#99–>

<!–T1:templates/html.jet#100–>
</FONT>
<!–T1:templates/html.jet#100–>

<!–T2:templates/html.jet#101–>
As maquinas de Turing deterministicas aceitam as mesmas linguagens que as MT n-deterministicas
<!–T2:templates/html.jet#101–>

<!–T1:templates/html.jet#102–>
<INPUT TYPE='RADIO' NAME='teste2' VALUE='Verdadeiro'>Verdadeiro
<INPUT TYPE='RADIO' NAME='teste2' VALUE='Falso'>Falso
<!–T1:templates/html.jet#102–>

<!–T5:templates/html.jet#97–>
<!– T7-Parent:templates/html.jet–>
<!–T7-Child:templates/alunos.jet#103–>

<!–T6:templates/html.jet#117–>

<!–T1:templates/alunos.jet#118–>
Alunos:
<!–T1:templates/alunos.jet#118–>

<!–T4:templates/alunos.jet#119–>

<!–T3:templates/alunos.jet#120–>
<!–T2:templates/alunos.jet#121–>
1234569
<!–T2:templates/alunos.jet#121–>
<!–T2:templates/alunos.jet#122–>

<!–T2:templates/alunos.jet#122–>
<!–T3:templates/alunos.jet#120–>

<!–T2:templates/alunos.jet#123–>
Fulano
<!–T2:templates/alunos.jet#123–>

<!–T1:templates/alunos.jet#124–>
 <!–T1:templates/alunos.jet#124–>

```
<!–T4:templates/alunos.jet#119–>
<!–T4:templates/alunos.jet#125–>

<!–T3:templates/alunos.jet#126–>
<!–T2:templates/alunos.jet#127–>
547862
<!–T2:templates/alunos.jet#127–>
<!–T2:templates/alunos.jet#128–>

<!–T2:templates/alunos.jet#128–>
<!–T3:templates/alunos.jet#126–>

<!–T2:templates/alunos.jet#129–>
Ciclano
<!–T2:templates/alunos.jet#129–>

<!–T1:templates/alunos.jet#130–>
<p> <!–T1:templates/alunos.jet#130–>

<!–T4:templates/alunos.jet#125–>
<!–T6:templates/html.jet#117–>

<!–T4:templates/html.jet#96–>
<!–T1:templates/html.jet#131–>
</body>
</html>
<!–T1:templates/html.jet#131–>
```

Arquivo *alunos.html*

```
<!–T1:templates/alunos.jet#104–>
Alunos:<p>
<!–T1:templates/alunos.jet#104–>

<!–T4:templates/alunos.jet#105–>

<!–T3:templates/alunos.jet#106–>
<!–T2:templates/alunos.jet#107–>
1234569
<!–T2:templates/alunos.jet#107–>
<!–T2:templates/alunos.jet#108–>

<!–T2:templates/alunos.jet#108–>
<!–T3:templates/alunos.jet#106–>

<!–T2:templates/alunos.jet#109–>
Fulano
```

```
<!–T2:templates/alunos.jet#109–>

<!–T1:templates/alunos.jet#110–>
<p>
<!–T1:templates/alunos.jet#110–>

<!–T4:templates/alunos.jet#105–>
<!–T4:templates/alunos.jet#111–>

<!–T3:templates/alunos.jet#112–>
<!–T2:templates/alunos.jet#113–>
547862
<!–T2:templates/alunos.jet#113–>
<!–T2:templates/alunos.jet#114–>

<!–T2:templates/alunos.jet#114–>
<!–T3:templates/alunos.jet#112–>

<!–T2:templates/alunos.jet#115–>
Ciclano
<!–T2:templates/alunos.jet#115–>

<!–T1:templates/alunos.jet#116–>
<p>
<!–T1:templates/alunos.jet#116–>

<!–T4:templates/alunos.jet#111–>
```

6 Anexo C

Resultados utilizando as *tags* originais JET.

Arquivo *html.jet*

```
<html>
<head>
<title>
<c:get select='$currHtml/@titulo' />
</title>
</head>
<body>
<c:iterate select='$currHtml/questao' var='t'>
<c:if test='$t/@num = '1"">
<FONT size='3' face='Tahoma' color='blue'>
<c:get select='$t/@tipo' />
</FONT> <p>
<c:get select='$t/enunciado' />
<p> <INPUT TYPE 'TEXT' NAME='nome' SIZE=50 VALUE=""> </p>
</c:if>
```

```
<c:if test='$t/@num = '2'">
<FONT size='3' face='Tahoma' color='blue'>
<c:get select='$t/@tipo' />
</FONT> <p>
<c:iterate select='$t/opcao' var='r'>
<INPUT TYPE='RADIO' NAME='teste1' VALUE='
<c:get select='$r/@id' /> '>
<c:get select='$r' />
<p>
</c:iterate>
</c:if>
<c:if test='$t/@num = '3'">
<FONT size='3' face='Tahoma' color='blue'>
<c:get select='$t/@tipo' />
</FONT> <p>
<c:get select='$t/enunciado' />
</p><INPUT TYPE='RADIO' NAME='teste2' VALUE='Verdadeiro'>Verdadeiro<p>
<INPUT TYPE='RADIO' NAME='teste2' VALUE='Falso'>Falso<p>
</c:if>
<ws:file template='templates/alunos.jet'
path='$org.eclipse.jet.resource.project.name/gerado/alunos.html' />
<c:include template='templates/alunos.jet'/>
</c:iterate>
</body>
</html>
```

Arquivo *alunos.jet*

```
Alunos:<p>
<c:iterate select='/questionario/aluno' var='p'>
<c:get select='$p/@ra' />
<c:get select='$p/nome' />
<p>
</c:iterate>
```

7 Anexo D

Resultados utilizando as *tags* originais JET.

Arquivo *teste.html*

```
<html>
<head>
<title>
Prova de LFA
</title> </head>
<body>

<FONT size="3"face="Tahoma"color="blue»
Dissertativa
```

</FONT>

Prove que todo automato finito det. é um NFA
 <INPUT TYPE "TEXT"NAME="nome"SIZE=50 VALUE=>
Alunos:

1234569
Fulano

547862
Ciclano

<FONT size="3"face="Tahoma"color="blue»
MultiplaEscolha
</FONT>
<INPUT TYPE="RADIO"NAME="teste1"VALUE="1 »
Toda linguagem regular é livre de contexto.

<INPUT TYPE="RADIO"NAME="teste1"VALUE="2 »
P = NP.

<INPUT TYPE="RADIO"NAME="teste1"VALUE="3 »
Não-Determinismo aumenta capacidade de reconhecimento de uma MT.

<INPUT TYPE="RADIO"NAME="teste1"VALUE="4 »
Nenhuma das anteriores

Alunos:
1234569
Fulano

547862
Ciclano

<FONT size="3"face="Tahoma"color="blue»
VerdadeiroOuFalso
</FONT>

As maquinas de Turing deterministicas aceitam as mesmas linguagens que as MT n-deterministicas

```
</p><INPUT TYPE="RADIO"NAME="teste2"VALUE="Verdadeiro»Verdadeiro<p>

<INPUT TYPE="RADIO"NAME="teste2"VALUE="Falso»Falso<p>

Alunos:<p>
1234569
Fulano
<p>
547862
Ciclano
<p>
</body>
</html>
```

Arquivo *alunos.html*

```
Alunos:<p>
1234569
Fulano
<p>

547862
Ciclano
<p>
```