Definição da Sintaxe Abstrata de Potigol em XML

Alessandro O. Anjos[1], Leonardo Lucena[2]

[1]Graduando em Tec em Análise e Desenvolvimento de Sistemas – IFRN. Bolsista do CNPq. E-mail: alessandro.anjos@academico.ifrn.edu.br
[2]Núcleo de Desenvolvimento de Software – IFRN. E-mail: leonardo.lucena@ifrn.edu.br

Resumo: O Núcleo de Desenvolvimento de Software do IFRN vem desenvolvendo a linguagem de programação Potigol para ser usada nos cursos introdutórios de computação. Além da linguagem em si, o projeto prevê o desenvolvimento de ferramentas de suporte. Essas ferramentas servirão, por exemplo, para analisar o código e verificar possíveis erros. Atualmente, a sintaxe concreta de Potigol ainda está em processo de aprimoramento, mas a sintaxe abstrata já está razoavelmente estável. Este artigo apresenta a definição da sintaxe abstrata de Potigol (PotiML) baseada na tecnologia XML. A sintaxe abstrata é definida através de dois recursos de XML: XML Schema e DTD. Como consequência, podemos prontamente integrar a linguagem Potigol com outras ferramentas, transformando o código dos programas para outras linguagens ou até criando programas diretamente na sintaxe abstrata.

Palavras–chave: ensino de programação, linguagens de programação, potigol, xml

1. INTRODUÇÃO

Um algoritmo pode ser escrito em qualquer linguagem de programação, mas a lógica é a mesma. De acordo com os recursos oferecidos pela linguagem o resultado final pode ser uma interface simples em modo de texto ou uma interface gráfica, ou até mesmo poderá ser mostrada em uma página da web e assim por diante [SAID, 2007]. Assim como nas línguas faladas por nós seres humanos as linguagens de programação possuem diferenças de sintaxe e de gramática, existem linguagens mais simples e outras mais difíceis, além de existirem linguagens que são mais adequadas de acordo com o tipo e da tarefa a ser realizada.

O desenvolvimento de algoritmos se dá por meio das linguagens de programação estruturada bem conhecidas como, por exemplo, as linguagens C, C++, Pascal, Fortran, entre outras, mas em nenhuma dessas linguagens existe um modelo de abstração que possa determinar que uma linguagem de programação possa ser codificada e transformada no que se refere a sintaxe e a gramática, e assim ter essa mesma sintaxe de um algoritmo em diferentes linguagens.

O objetivo da linguagem PotiML é que ela seja uma linguagem baseada na tecnologia XML e que essa mesma linguagem também seja uma definição abstrata da linguagem Potigol e que apresente os mesmos elementos de uma linguagem de programação de alto nível porém em XML e assim permitindo que os algoritmos escritos em PotiML sejam transformados facilmente em Potigol ou em outras linguagens de programação.

Este trabalho está inserido em um projeto de pesquisa para a construção de uma linguagem no formato da tecnologia XML desenvolvida em conjunto com a linguagem de programação Potigol e que possibilitará a construção e a definição abstrata dessa linguagem. A primeira etapa deste projeto maior é a avaliação da construção da linguagem Potiml. Este processo de avaliação está divido em duas etapas: uma com a avaliação da construção de uma sintaxe sendo construída por meio do DTD e do Schema e posteriormente a construção de um reconhecedor de sentenças (*parser)* utilizando a linguagem Scala ou Java. Assim, no trabalho apresentado iremos mostrar os resultados obtidos na avaliação e no desenvolvimento de uma sintaxe utilizando o DTD e o Schema com as mesmas construções da linguagem Potigol.

ISBN 978-85-62830-10-5

2. A LINGUAGEM XML

A Linguagem **XML** (eXtensible Markup Language) - mantida pelo comitê do consórcio internacional W3C (World Wide Web Consortium), responsável pelos padrões e diretrizes usados na Web [W3C, 2012] – foi definida com os seguintes objetivos:

- Descrever dados, dando semântica (significado) a unidade de informação;
- Trocar dados na Internet ou qualquer meio eletrônico (*transferência eletrônica entre banco de dados, correio eletrônico, etc.*);
- Armazenar e organizar documentos (*armazenar o conteúdo de documentos em formato aberto*);
- Descrever metadados de uma aplicação.

No XML o desenvolvedor tem a liberdade de escolher o nome que lhe convier.

Com os dados XML válidos e bem-formatados o documento XML se torna autodescritivo porque as *tags* dão a ideia de conteúdo e são misturadas com os dados (Figura 1). Como o formato do documento é aberto e flexível ele pode ser usado em qualquer lugar onde a troca ou a transferência da informação seja necessária. Com isto podemos usar o XML para descrever informações sobre páginas HTML ou descrever dados contidos em objetos, ou também em regras de negócios, transações eletrônicas, comerciais, etc.

A linguagem XML é extensível por isso permite que o desenvolvedor crie as suas próprias *tags* livremente. Entretanto, esta liberdade pode gerar conflitos como, por exemplo, em um mesmo documento poderão ser apresentados elementos distintos e de ordens variadas, por isso existem linguagens de definição de esquema XML. Estes documentos informam quais são os seus elementos, atributos e a ordem destes. Dentre estas linguagens destacam-se o DTD e o XML Schema [LOSCIO, 2009], sendo este último recomendado pelo W3C por possuir uma melhor definição dos dados e por ser escrita em XML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Pessoa cpf="123.465.789-10">
 <DadosPessoais>
  <nome>Juliana Marques</nome>
  <apelido>Juju</apelido>
  <naturalidade>Natal, RN</naturalidade>
 </DadosPessoais>
</Pessoa>
```

Figura 1 – Exemplo de um trecho de código no formato da tecnologia da linguagem XML.

3. O DTD

O DTD (*Document Type Definition*) define um padrão para marcação de dados em documentos por meio da definição de uma hierarquia de elementos, onde a partir de um elemento raiz desta hierarquia há outros elementos compostos por atributos, entidades, subelementos e comentários. Para que um documento XML esteja de acordo com o DTD apenas os elementos e as estruturas de alinhamento entre os elementos definidos no DTD são permitidos no corpo do documento, esta validação é feita por um *parser* XML.

O DTD pode tanto ser definido no próprio documento XML como também em um arquivo externo. Um arquivo DTD e todos os documentos XML (e documentos HTML) são compostos por blocos de construção como os:

ISBN 978-85-62830-10-5

- Elementos
- Atributos
- PCDATA
- CDATA

Os Elementos são os principais blocos de construção de documentos XML (Figura 2). Os Atributos fornecem as informações adicionais aos elementos. PCDATA e CDATA são textos que compõe o documento XML sendo que o primeiro o PCDATA será analisado por um *parser*. O CDATA significa dados de caracteres. O CDATA é utilizado nos atributos e o PCDATA nos elementos do DTD.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE Pessoa [
 <!ELEMENT Pessoa (DadosPessoais)>
 <!ATTLIST Pessoa cpf CDATA>
 <!ELEMENT DadosPessoais (nome, apelido, naturalidade)>
 <!ELEMENT nome (#PCDATA)>
 <!ELEMENT apelido (#PCDATA)>
 <!ELEMENT naturalidade (#PCDATA)>
]>
<Pessoa cpf="123.465.789-10">
 <DadosPessoais>
  <nome>Juliana Marques</nome>
  <apelido>Juju</apelido>
  <naturalidade>Natal, RN</naturalidade>
 </DadosPessoais>
</Pessoa>
```

Figura 2 – Exemplo de um trecho de código de uma declaração interna da linguagem XML utilizando o DTD.

Um DTD pode definir a estrutura básica de um documento XML, no entanto, o DTD possui limitações na sua construção tornando uma linguagem limitada no seu poder de expressão. Uma dessas limitações é de não ser possível restringir o tipo de conteúdo de um elemento ou de um atributo. Não é possível, por exemplo, estabelecer que o valor de um determinado atributo seja um valor numérico e não um texto qualquer, além disso, o DTD possui tipos de dados limitados. O W3C para recuperar as limitações do DTD definiu a linguagem XML Schema como sendo uma linguagem de definição de sintaxe em XML.

4. O SCHEMA

Como alternativa ao DTD surge o XML Schema (Figura 3) que é um método utilizado para especificar uma gramática ou um apanhado de regras para um conjunto de documentos XML. O XML Schema tem o objetivo de fornecer uma linguagem mais poderosa no que diz respeito na especificação de gramáticas para documentos XML. O XML Schema da W3C é o mais promissor para substituir o DTD, pois define:

- Elementos e atributos em XML;
- Define a ordem e a hierarquia dos elementos;
- Define o tipo de dados dos elementos e dos atributos;
- Define valores default e valores fixos para os elementos e os atributos.

ISBN 978-85-62830-10-5

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
 xmlns=http://www.w3schools.com targetNamespace=http://www.w3schools.com
xmlns:xs="http://www.w3.org/2001/XMLSchema"  elementFormDefault="qualified"
attributeFormDefault="unqualified">
 <xs:element name="Pessoa">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="DadosPessoais"/>
   </xs:sequence>
   <xs:attribute name="cpf" type="xs:string"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="DadosPessoais">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="nome"/>
    <xs:element ref="apelido"/>
    <xs:element ref="naturalidade"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="nome" type="xs:string"/>
 <xs:element name="apelido" type="xs:string"/>
 <xs:element name="naturalidade" type="xs:string"/>
</xs:schema>

<Pessoa cpf="123.465.789-10">
 <DadosPessoais>
  <nome>Juliana Marques</nome>
  <apelido>Juju</apelido>
  <naturalidade>Natal, RN</naturalidade>
 </DadosPessoais>
</Pessoa>
```

Figura 3 – Exemplo de um trecho de código de uma declaração interna da linguagem XML utilizando o Schema.

5. POTIGOL

A linguagem Potigol [LUCENA, 2011] está sendo elaborada com o objetivo ser uma linguagem de programação voltada para brasileiros que estão aprendendo programação. Potigol é uma linguagem multiparadigma que mistura elementos das linguagens Scala [ODERSKY, 2010], Ruby [FLAMAGAN, 2008], Portugol [SOUSA, 2009] e Java [DEITEL, 2003] que equivalha a mesma lógica de programação de alto nível das outras linguagens de programação conhecidas para resolver as dúvidas e a soluções no desenvolvimento de algoritmos fase em que o iniciante necessita ver o resultado da sequência lógica das instruções que compiladas, editadas e executadas irá compor o algoritmo. Para alcançar o objetivo está sendo formulada uma maneira fácil e prática para o aluno trabalhar com a linguagem Potigol (Figura 4), pois o código deve-se tornar o mais enxuto, prático e objetivo possível.

6. POTIML

O XML é considerado como ser uma metalinguagem por ser uma linguagem capaz de gerar outras linguagens, para isso utiliza-se o DTD ou o Schema em que esses são definidos especificando quais *tags* podemos ou deveremos encontrar nos documentos XML, incluindo a ordem e descrevendo

ISBN 978-85-62830-10-5

os atributos que poderão ou deverão estar contidos em todas as *tags*. Assim, o documento XML pode ter as suas próprias *tags* (definidas em um DTD ou no Schema) do mesmo modo como uma linguagem de programação tem as suas palavras chaves.

A linguagem PotiML surge para ser uma linguagem que seja baseada na tecnologia XML e também que essa seja uma linguagem abstrata da linguagem Potigol, para isso deve apresentar os mesmos recursos de uma linguagem de programação de alto nível em XML e permitir que os algoritmos escritos em PotiML sejam transformados em Potigol ou em outras linguagens de programação.

```
programa ranking
  tipo Participante(nome do tipo Texto, nota do tipo Inteiro)
    função dados = nome + " - " + nota
  fim
  seja pessoas =
    para i em [1 .. 200] gere
  //Lista.preencher(200)
      escreva "Digite o nome: "
      seja nome = lerTexto()
      escreva "Digite a nota: "
      seja nota = lerInteiro()
      Participante(nome, nota)
    fim
  escreva pessoas
      .filtrar(_.nota > 70)
      .ordenarUsando(_.nota)
      .reverter
      .mapear(_.dados)
      .comoTexto("\n")
fim
```

Figura 4 – Exemplo de um programa escrito na linguagem de programação multiparadigma Potigol.

Para a definição da sua sintaxe e do esquema da linguagem PotiML é utilizado o DTD e o XML Schema estes informam quais são os elementos, os atributos e a ordem destes para serem utilizados na linguagem PotiML como pode ser observado na Figura 5.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://www.w3schools.com"
  targetNamespace="http://www.w3schools.com"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="programa">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="atribuicao"/>
        <xs:element ref="escrever"/>
        <xs:element ref="funcao"/>
        <xs:element ref="valor"/>
        <xs:element ref="para"/>
        <xs:element ref="enquanto"/>
        <xs:element ref="variavel"/>
      </xs:choice>
      <xs:attribute name="nome" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="escrever">
    <xs:complexType mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="exp"/>
      </xs:choice>
      <xs:attribute name="linha" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="exp">
    <xs:complexType>
      <xs:attribute name="valor" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="funcao">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="parametros"/>
        <xs:element ref="exp"/>
      </xs:sequence>
      <xs:attribute name="nome" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="parametros">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="parametro" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="parametro">
    <xs:complexType>
      <xs:attribute name="nome" type="xs:string" use="required"/>
      <xs:attribute name="tipo" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="valor">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="casamento"/>
        <xs:element ref="exp"/>
        <xs:element ref="ler"/>
        <xs:element ref="se"/>
      </xs:choice>
      <xs:attribute name="nome" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
```

Figura 5 – Trecho do XML Schema para a definição da sintaxe e para o uso da linguagem PotiML.

ISBN 978-85-62830-10-5

Além de definir a linguagem PotiML pelo uso do XML Schema também é possível defini-la pelo DTD como é mostrado na Figura 6.

```
<!ELEMENT programa ( escrever | funcao | valor
| para | enquanto | variavel | atribuicao )* >
<!ATTLIST programa nome NMTOKEN #REQUIRED >
<!ELEMENT escrever ( #PCDATA | exp )* >
<!ATTLIST escrever linha NMTOKEN #IMPLIED >
<!ELEMENT exp EMPTY >
<!ATTLIST exp valor CDATA #REQUIRED >
<!ELEMENT funcao ( parametros, exp ) >
<!ATTLIST funcao nome NMTOKEN #REQUIRED >
<!ELEMENT parametros ( parametro+ ) >
<!ELEMENT parametro EMPTY >
<!ATTLIST parametro nome NMTOKEN #REQUIRED >
<!ATTLIST parametro tipo NMTOKEN #REQUIRED >
<!ELEMENT valor ( casamento | exp | ler | se
| atribuicao )* >
<!ATTLIST valor nome CDATA #REQUIRED >
<!ELEMENT casamento ( caso+ ) >
<!ATTLIST casamento exp NMTOKEN #REQUIRED >
<!ELEMENT caso ( #PCDATA | padrao )* >
<!ELEMENT padrao ( #PCDATA ) >
<!ATTLIST caso padrao NMTOKEN #IMPLIED >
<!ELEMENT se ( entao, senao ) >
<!ATTLIST se condicao CDATA #REQUIRED >
<!ELEMENT entao ( #PCDATA ) >
<!ELEMENT senao ( #PCDATA ) >
```

```
<!ELEMENT ler EMPTY >
<!ATTLIST ler tipo NMTOKEN "Texto" >
<!ELEMENT atribuicao (exp)>
<!ATTLIST atribuicao var CDATA #REQUIRED>
<!ELEMENT para ( atribuicao | escrever | para
| valor | enquanto | variavel )* >
<!ATTLIST para
   var CDATA #REQUIRED
   inicio NMTOKEN #REQUIRED
   fim CDATA #REQUIRED >
<!ELEMENT enquanto ( atribuicao | escrever
| para | valor | enquanto | variavel )* >
<!ATTLIST enquanto
         fim CDATA #REQUIRED>
<!ELEMENT variavel EMPTY>
<!ATTLIST variavel
         nome CDATA #REQUIRED
         tipo CDATA #REQUIRED
         valor CDATA #IMPLIED>
```

Figura 6 – Trecho do DTD para definição da sintaxe e para o uso da linguagem PotiML.

Com a construção para a definição da sintaxe da linguagem PotiML utilizando o XML Schema e o DTD é possível desenvolver algoritmos com uma coleção de *tags* e aplica-las na linguagem através de uma referência do DTD ou do Schema e assim ficar livre para poder escrever os algoritmos como é utilizado nas Figuras 7 e 8.

```
<?xml version="1.0" encoding="utf-8"?>

<programa nome="HelloWorldPotiML"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="Potiml.xsd">

   <escrever> Olá Mundo </escrever>

</programa>
```

Figura 7 – Programa escrito em PotiML referenciado por um XML Schema.

ISBN 978-85-62830-10-5

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE programa SYSTEM "Potiml.dtd">

<programa nome="soma">

  <variavel nome="a" tipo="Inteiro" valor="10" >
  <variavel nome="b" tipo="Inteiro" valor="10" >

  <valor nome="soma">
      <exp valor="a+b" />
  </valor>

  <escrever>
     A soma de <exp valor="a" />
               e <exp valor="b" />
      é de: <exp valor="soma" />
  </escrever>

</programa>
```

Figura 8 – Programa escrito em PotiML referenciado por um DTD através da diretiva <!DOCTYPE>.

Como temos a definição e a estrutura da linguagem desenvolvida agora o desenvolvimento de programas e algoritmos em PotiML torna-se possível e interessante a partir do momento que o desenvolvedor escreve e deseja completar um trecho de código do algoritmo, isso será possível quando o próprio desenvolvedor pressionar as teclas Ctrl + Space e daí será fornecido uma lista de opções para escrever e completar a *tag* que o desenvolvedor queria tudo isso para facilitar e ajudar o usuário no desenvolvimento como pode ser observado na Figura 9.

Figura 9 – Lista de opções fornecidas para ajudar no desenvolvimento do algoritmo na linguagem PotiML.

Para desenvolver na linguagem PotiML poderá ser utilizado a IDE (Integrated Development Environment) ou Ambiente de Desenvolvimento Integrado utilizando o Eclipse ou um simples editor de texto comum e assim o usuário estará livre para desenvolver qualquer programa. A linguagem não é *case sesitive*, ou seja, não faz diferença entre letras maiúsculas e minúsculas o que torna uma linguagem de fácil uso.

7. TRADUÇÃO DA LINGUAGEM POTIGOL PARA POTIML OU VICE-VERSA

ISBN 978-85-62830-10-5

A linguagem PotiML pode ser traduzida para uma linguagem de programação qualquer como pode ser observado na Tabela 1.

Tabela 1 - Exemplo da tradução de um algoritmo escrito na linguagem PotiML convertido para o Potigol.

```
<!--PotiML -->
<programa nome="Situação">
  <escrever>Digite a sua média:</escrever>
    <valor nome="média">
      <ler tipo="Inteiro" />
    </valor>
  <valor nome="situação">
    <se condição="média &gt;70">
      <então>Aprovado</então>
      <senão>Reprovado</senão>
    </se>
  </valor>
  <escrever> A sua situação é
    <exp valor="situação" />
  </escrever>
</programa>
```

```
# Potigol
programa Situação
  escreva "Digite a sua média"
  média = leiaInteiro
  situação =
    se média > 70 então
      "Aprovado"
    senão
      "Reprovado"
    fim
    escreva "A sua situação é " + situação
fim
```

No Potigol e na maioria das linguagens de programação são utilizados diferentes formatos de iniciar e fechar trechos em um determinado código como nos métodos e nas funções. Isso com a abertura e o fechamento de uma chave começando com **"{"** e terminando com **"}"** ou como no Potigol que se inicia um trecho de código com a palavra reservada "***programa"*** e termina com a palavra reservada "***fim"***. Entretanto, isso não é possível no PotiML e nem nas linguagens que usam a tecnologia do XML, pois o início e o fim de uma *tag* deverão ter os mesmos nomes como foi mostrado no exemplo acima. Nesse mesmo exemplo pode-se observar na primeira linha um trecho de comentário sendo utilizado.

8. CONCLUSÕES E TRABALHOS FUTUROS

Diante da importância e do nível de dificuldade das disciplinas de programação aprender uma nova linguagem como a PotiML e coloca-la em prática é sem dúvida uma nova forma de reduzir o obstáculo a ser superado no aprendizado de programação.

Por ser uma ferramenta de fácil utilização que permite construir pequenos algoritmos sem a necessidade da construção de todas as suas etapas complexas a linguagem também agrega grande potencial ao dia-a-dia dos profissionais de informática envoltos do ambiente de programação e mais precisamente em um ambiente de programação web.

Uma função que a linguagem PotiML poderá fornecer e possibilitar no aprendizado sendo assim como proposta de trabalho futuro é demonstrando que a mesma sintaxe poderá ser transformada em outras linguagens de programação através da construção de um reconhecedor de sentenças permitindo que o algoritmo escrito em PotiML possa ser transformado em outra linguagem de programação desejada como o Potigol.

REFERÊNCIAS

CARRO, S. **Curso de XML Completo.** Disponível em: <http://pt.scribd.com/doc/83565628/Curso-de-XML-completo> Acesso em: 02 jul 2012.

DEITEL, H.; DEITEL, P. **Java como programar**. Porto Alegre: Bookman, 2003.

FLAMAGAN, D. **Linguagem de Programação Ruby.** Altas Books, 2008.

ISBN 978-85-62830-10-5

LÓSCIO, B; FILHO, F. **Web Semântica: Conceitos e Tecnologias** - UNIVERSIDADE FEDERAL DO PIAUÍ, Capitulo 9, 2009.

LUCENA, L. **A Linguagem Potigol.** Relatório Técnico. Disponível em: <https://docs.google.com/document/d/1HqFlvoV6un_XZTMK8zoBctexgeJQxTp1wx3dQAUYKDk/edit> Acesso em: 27 jun 2012.

ODERSKY, M; SPOON, L; VENNERS, B. **Programming in Scala, Second Edition: A Comprehensive Step-by-Step Guide**. Artima Press, 2010.

SAID, R. **Curso de Lógica de Programação**. São Paulo: Digerati Books, 2007.

SOUSA, C. **VisuAlg - Ferramenta de Apoio ao Ensino de Programação**. Vol. 2, Revista TECCEN, 2009, pp. 1-9.

W3C. **World Wide Web Consortium.** Disponível em: <http://www.w3.org> Acesso em: 27 jun 2012.

ISBN 978-85-62830-10-5