Comunicado Técnico 63

Novembro, 2004
Campinas, SP

ISSN 1677-8464

Adaptação da Ferramenta Digester para Tratamento de Atributos em *Tags* de Documentos XML

Sérgio Aparecido Braga da Cruz[1]
Carla Geovana do Nascimento Macário[2]

A linguagem XML - eXtensible Markup Language (World Wide Web Consortium, 2004a), uma das tecnologias evidenciadas com o surgimento da WWW (World Wide Web) (Krol, 1993), é um padrão para formatação de textos criado com o objetivo inicial de facilitar a publicação eletrônica de conteúdo na WWW. A flexibilidade deste padrão fez com que ele fosse gradualmente utilizado em outras áreas, sendo que atualmente possui papel importante também na troca de informações entre aplicações, tanto na WWW quanto fora dela. Com isso e estimulados pelo fato deste ser um formato não-proprietário e um padrão recomendado pela W3C (WWW Consortium), principal organismo responsável pela definição de padrões para a WWW, várias ferramentas surgiram para apoiar o uso da XML.

O projeto Agência de Informação desenvolvido pela Embrapa Informática Agropecuária utiliza a tecnologia XML na troca de dados entre alguns de seus módulos e adota a ferramenta Digester (Apache Software Foundation, 2004d) para processamento destes dados. Esta ferramenta apresentou algumas limitações quando utilizada no tratamento de dados em formato XML, estruturados de acordo com as necessidades do Projeto Agência. Este documento descreve as adaptações implementadas na ferramenta Digester permitindo contornar estas limitações.

Padrão XML

A linguagem XML é um subconjunto do padrão SGML - Standard Generalized Markup Language (University of Virginia, 2004) para representação de texto em meio eletrônico. O termo *Markup* da sigla SGML originou-se da tipografia e é utilizado para descrever as marcações realizadas num texto a ser impresso indicando ao tipógrafo as mudanças de tipos de impressão. Este termo permaneceu após a automação do processo de impressão e descreve os mecanismos utilizados para indicar tipos de formatação do texto em meio eletrônico. Uma evolução do uso da marcação nos meios eletrônicos foi a sua utilização para indicar também conceitos em documentos. Implicitamente os documentos podem indicar, por meio de sua organização, alguns elementos principais tais como títulos, parágrafos, items, subitems, etc., os quais auxiliam o entendimento e uma melhor organização do conteúdo de um documento. A marcação em meio eletrônico torna explícita a delimitação destes elementos e permite que novos elementos sejam demarcados podendo levar em conta também a sua perspectiva conceitual. Uma linguagem de marcações, como a SGML, define uma sintaxe para especificação das marcações, as quais podem ser interpretadas tanto do ponto

[1] *M.Sc. em Engenharia Elétrica, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: sergio@cnptia.embrapa.br)*
[2] *M.Sc. em Engenharia Elétrica, Pesquisadora da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: carla@cnptia.embrapa.br)*

de vista de formatação de texto quanto do ponto de vista conceitual do texto demarcado (University of Virginia, 2004). Uma característica fundamental da SGML que tornou o seu uso tão flexível é que, assim como o texto demarcado, as marcações também são especificadas na forma de texto. Assim a linguagem de marcações define uma metalinguagem legível para o ser humano sobre o texto original. Arquivos SGML que foram formatados em 1985, época da criação deste padrão, permanecem acessíveis ainda hoje, o que raramente aconteceu com formatos proprietários. Outras características da linguagem SGML que a torna diferente de outras linguagens de marcação são a ênfase no caráter descritivo da marcação, a classificação de tipos de documentos de acordo com a sua estrutura e a utilização de um mecanismo de substituição de cadeias de caracteres, o que facilita o tratamento de um mesmo caracter em diferentes ambientes computacionais.

A linguagem XML surgiu como uma iniciativa do W3C em meados de 1996 com o objetivo de ser uma versão simplificada da SGML e possibilitando que as qualidades desta linguagem pudessem ser aproveitadas na WWW. Para tanto a linguagem XML foi projetada para ser facilmente processada e integrada tanto com a SGML quanto com o HTML (versões 3.2 em diante) (World Wide Web Consortium, 2004b), outra linguagem surgida com a WWW que permite a formatação de documentos hipertexto envolvendo recursos multimídia.

A linguagem XML herdou várias características do SGML, principalmente a sua portabilidade sem perda de informação para diferentes ambientes computacionais, uma necessidade no ambiente de troca de documentos proporcionado pela WWW. As marcações em XML são textos conhecidos como *tags* que demarcam os limites do texto marcado. As *tags* seguem a forma "<" *texto* ">" para indicar o início da marcação e "</" *texto* ">" para indicar o fim da marcação. Na *tag* inicial podem aparecer termos na forma *nome*="*valor*" indicando algum atributo da *tag* (Mordani & Boag, 2002). A Fig. 1 apresenta um exemplo de arquivo no formato XML. Este exemplo descreve dados sobre um livro.

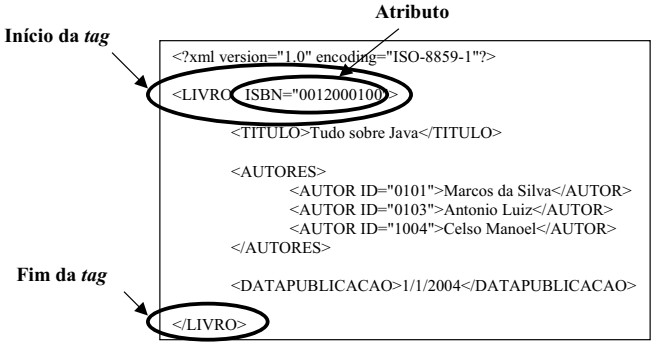

FIG. 1. Exemplo de um arquivo XML.

Na primeira linha deste exemplo temos uma declaração do documento XML indicando a versão e o padrão de codificação de caracteres utilizado no documento, que neste caso são respectivamente versão 1.0 e padrão de codificação ISO-8859-1. *LIVRO*, *TITULO*, *AUTORES*, *AUTOR* e *DATAPUBLICACAO* são as *tags* utilizadas neste documento. *ISBN* é um atributo da *tag LIVRO* e *ID* um atributo da *tag AUTOR*.

Processamento de Documentos XML

Uma ferramenta para processamento de documentos XML que se tornou um padrão pela sua grande utilização é a SAX - Simple API for XML (SAX, 2004), uma biblioteca inicialmente implementada em Java e agora disponível para várias outras linguagens, que permite o processamento de documentos XML. A SAX implementa uma abordagem orientada a eventos para sinalizar, durante o processamento do documento, a identificação das *tags* e de outros elementos estruturais de documentos XML. Estes eventos são encaminhados para o resto do programa Java, que poderá então tratá-lo da maneira mais adequada. A construção da parte relativa ao tratamento adequado dos eventos corresponde à implementação de uma classe tratadora de eventos ou *handler*. Apesar de ser simples como o próprio nome indica, a implementação de soluções de processamento de documentos XML diretamente sobre a biblioteca SAX exige um certo esforço de programação, particular para cada tipo de documento que se pretende processar.

Na Agência, os documentos XML estão sendo utilizados como mecanismo de troca de dados estruturados entre formulários da interface com o usuário e o sistema. Estes dados estruturados utilizando XML são mais facilmente processados para uso posterior de seu conteúdo dentro do sistema. A ferramenta Digester se mostrou como uma solução de mais alto nível e genérica, em relação à biblioteca SAX, para processamento de documentos XML de acordo com as necessidades da Agência.

Ferramenta Digester

A ferramenta Digester (Apache Software Foundation, 2004d) é um dos componentes gerados pelo subprojeto Jakarta Commons (Apache Software Foundation, 2004a), o qual é um dos subprojetos do Projeto Jakarta (Apache Software Foundation, 2004b), que tem como objetivo implementar soluções *open source* utilizando a linguagem Java. Esta ferramenta foi inicialmente criada como um mecanismo para carga do arquivo de configuração da *framework* para automação do fluxo de controle de aplicações WWW chamada *Struts* (Apache Software Foundation, 2004c). A simplicidade e a facilidade de uso da solução fez com que ela fosse amplamente adotada por diversos outros projetos e sistemas de tal forma que foi destacada do *Struts* e passou a ser distribuída como um componente separado. A Digester implementa um

mecanismo de amarração entre um documento XML e um objeto Java, ou seja, permite que os dados de um documento XML sejam adequadamente armazenados num objeto Java.

Para realizar esta tarefa a Digester utiliza a biblioteca SAX como base, e sobre ela, um conjunto de classes que facilitam a sua utilização. Estas classes formam um mecanismo genérico para tratamento dos eventos gerados pelo SAX e permitem que para cada evento gerado seja mapeado uma ação de execução de um método ou de criação de um objeto Java. Estas ações permitem o armazenamento da informação associada ao evento no objeto Java. Cada evento é gerado pela SAX quando são identificadas várias situações relativas à estrutura do documento XML durante o seu processamento.

O modo como os eventos devem ser tratados é fornecido para a Digester como um atributo organizado na forma de um conjunto de regras para processamento dos eventos. Ao ser requisitado para tratar um evento, a Digester busca neste conjunto de regras aquela mais adequada para tratamento do evento e dispara a execução de processamento associada. As regras fazem o mapeamento entre os eventos e o código da aplicação que deverá ser executado. O diagrama da Fig. 2 ilustra o funcionamento da Digester.

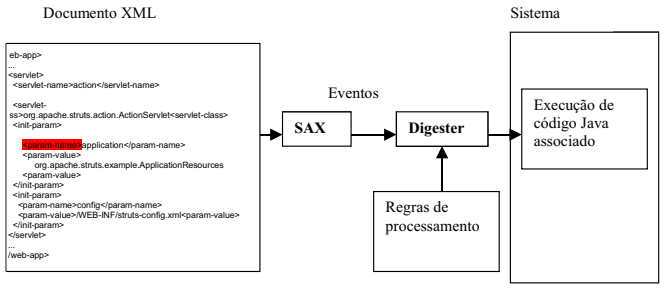

FIG. 2. Funcionamento geral da Digester.

Mapeamento de eventos de Processamento XML

A estrutura das regras que definem o mapeamento entre os eventos e as ações a serem executadas pode ser dividida em duas partes: descrição da *tag* associada ao evento e especificação do código a ser executado.

As *tags* são especificadas usando uma notação semelhante à notação de diretórios, a qual reflete a sua estrutura hierárquica. Esta notação define um padrão para a comparação com a *tag* que serve de contexto na ocorrência do evento. Se a *tag* "casar" com o padrão, a ação associada à regra é executada. A Fig. 3 apresenta um documento XML simples e os padrões correspondentes.

Existem dois tipos de ações na Digester. Um deles é relacionado à criação de novos objetos e o outro relacionado à chamada de métodos sobre os objetos criados. Todo objeto criado pela Digester é armazenado numa pilha interna de objetos Java e a invocação de métodos refere-se à chamada de métodos dos objetos armazenados nesta pilha.

XML	Padrão
<a>	--> “a”
	--> “a/b”
<c/>	--> “a/b/c”
	--> “a/b”
<c/>	--> “a//b/c”
<c/>	--> “a/b/c”
	

FIG. 3. Padrões associados às *tags* XML.

As regras disponíveis na Digester não permitem um acesso aleatório aos elementos da pilha, fornecendo acesso somente ao elemento raiz ou inicial criado na pilha, ou aos elementos próximos ao elemento topo da pilha, de maneira relativa. Alguns exemplos de regras implementadas na Digester são:

- ObjectCreateRule - cria novo objeto e coloca-o no topo da pilha.
- SetNextRule - invoca o método do objeto imediatamente anterior ao topo da pilha passando como argumento o objeto no topo da pilha.
- SetTopRule - invoca o método do objeto no topo da pilha passando como argumento o objeto imediatamente anterior ao topo da pilha.
- SetRootRule - invoca o método do objeto raiz ou inicial da pilha passando como argumento o objeto no topo da pilha.
- SetPropertyRule - atribui valor da propriedade do objeto no topo da pilha, obtido da propriedade de mesmo nome existente em *tag* XML.
- CallMethodRule - invoca o método de objeto no topo da pilha, passando como argumento valores especificados pela regra CallParamRule.
- CallParamRule - especifica valores a serem utilizados por um CallMethodRule.
- BeanPropertySetterRule - atribui valor de propriedade do objeto no topo da pilha obtido da parte textual demarcada por uma tag XML com nome correspondente.

Um exemplo completo de uso da Digester é apresentado na Fig. 4. A Fig. 4(a) mostra um documento XML simples contendo dados sobre um livro. Na Fig. 4(b), estão as implementações simplificadas das classes Livro e Autor em linguagem Java, criadas para armazenar as informações relativas a um livro e seus autores. A Fig. 4(c) apresenta o código principal, que implementa o processamento do documento XML permitindo a extração dos dados do documento e seu armazenamento na estrutura dos objetos Java.

```
<LIVRO ISBN="0012000100">
        <TITULO>Tudo sobre Java</TITULO>
        <AUTORES>
                <AUTOR ID="0101">Marcos da Silva</AUTOR>
                <AUTOR ID="0103">Antonio Luiz</AUTOR>
                <AUTOR ID="1004">Celso Manoel</AUTOR>
        </AUTORES>
        <DATAPUBLICACAO>1/1/2004</DATAPUBLICACAO>
</LIVRO>
```

(a)

```
public class Livro {
        private String isbn;
        private String titulo;
        private Collection autores;
        private String datapublicacao;

        //set´s e get´s para os atributos da classe
        public String toString() { ... }
}

public class Autor {
        private String id;
        private String nome;
        //set´s e gets para os atributos da classe
}
```

(b)

```
import org.apache.commons.digester.Digester;
import org.xml.sax.SAXException;
import java.util.ArrayList;
import java.io.File;
import java.io.IOException;

public class LivroDigester {

   public static void main(String [] args) throws IOException, SAXException
   {
      Digester digester = new Digester();

      digester.addObjectCreate("LIVRO", Livro.class);
      digester.addSetProperties("LIVRO","isbn","isbn");
      digester.addBeanPropertySetter("LIVRO/TITULO","titulo");
      digester.addBeanPropertySetter("LIVRO/DATAPUBLICACAO","dataPublicacao");
      digester.addObjectCreate("LIVRO/AUTORES", ArrayList.class);
      digester.addObjectCreate("LIVRO/AUTORES/AUTOR",Autor.class);
      digester.addSetNext("LIVRO/AUTORES/AUTOR","add");
      digester.addSetProperties("LIVRO/AUTORES/AUTOR","ID","id");
      digester.addBeanPropertySetter("LIVRO/AUTORES/AUTOR","nome");
      digester.addSetNext("LIVRO/AUTORES","setAutores");

      Livro livro = (Livro) digester.parse(new File("livro.xml"));
      System.out.println(livro);
   }
}
```

(c)

FIG. 4. Exemplo de uso da Digester (a) Documento em formato XML; (b) Código Java simplificado; (c) Código Java para processamento do documento XML.

Os passos seguidos no processamento do documento XML pela Digester para o exemplo apresentado na Fig. 4 são os apresentados a seguir e graficamente na Fig. 5:

1. Um objeto da classe Livro é criado na pilha ao ser encontrada a *tag* LIVRO;

   1.1. O valor da propriedade *isbn* do objeto livro é obtido do atributo ISBN da *tag* LIVRO;

   1.2. O valor da propriedade *titulo* do objeto livro é obtido do texto demarcado pela *tag* TITULO, dentro da *tag* LIVRO (padrão "LIVRO/TITULO");

   1.3. O valor da propriedade *datapublicacao* é obtido do texto demarcado pela *tag* DATAPUBLICACAO, dentro da *tag* LIVRO (padrão "LIVRO/DATAPUBLICACAO");

2. Um objeto da classe ArrayList é criado na pilha ao ser encontrada a *tag* AUTORES;

   2.1. A propriedade autores de livro recebe uma referência para esta lista através de chamada de método *add*;

3. Um objeto da classe Autor é criado na pilha ao ser encontrada a *tag* AUTOR;

3.1. O objeto da classe Autor é adicionado na lista através de chamada de método;

3.2. O valor da propriedade *id* do objeto autor é obtido do atributo ID da *tag* AUTOR;

3.3. O valor da propriedade *nome* do objeto autor é obtido do texto demarcado pela *tag* AUTOR;

4. O objeto da classe Autor é desempilhado quando é encontrada </AUTOR>, uma referência ao objeto criado permanece em ArrayList;

5. Um objeto da classe Autor é criado na pilha ao ser encontrada a segunda *tag* AUTOR;

6. O objeto da classe Autor é desempilhado quando é encontrada </AUTOR>, uma referência ao objeto criado permanece em ArrayList;

7. Um objeto da classe Autor é criado na pilha ao ser encontrada a terceira *tag* AUTOR;

8. O objeto da classe Autor é desempilhado quando é encontrada </AUTOR>, uma referência ao objeto criado permanece em ArrayList;

9. O objeto da classe ArrayList é retirado da pilha, uma referência a este objeto permanece armazenada no atributo autores do objeto da classe Livro.

10. Fim de execução do parser, objeto livro é retornado com atributos setados.

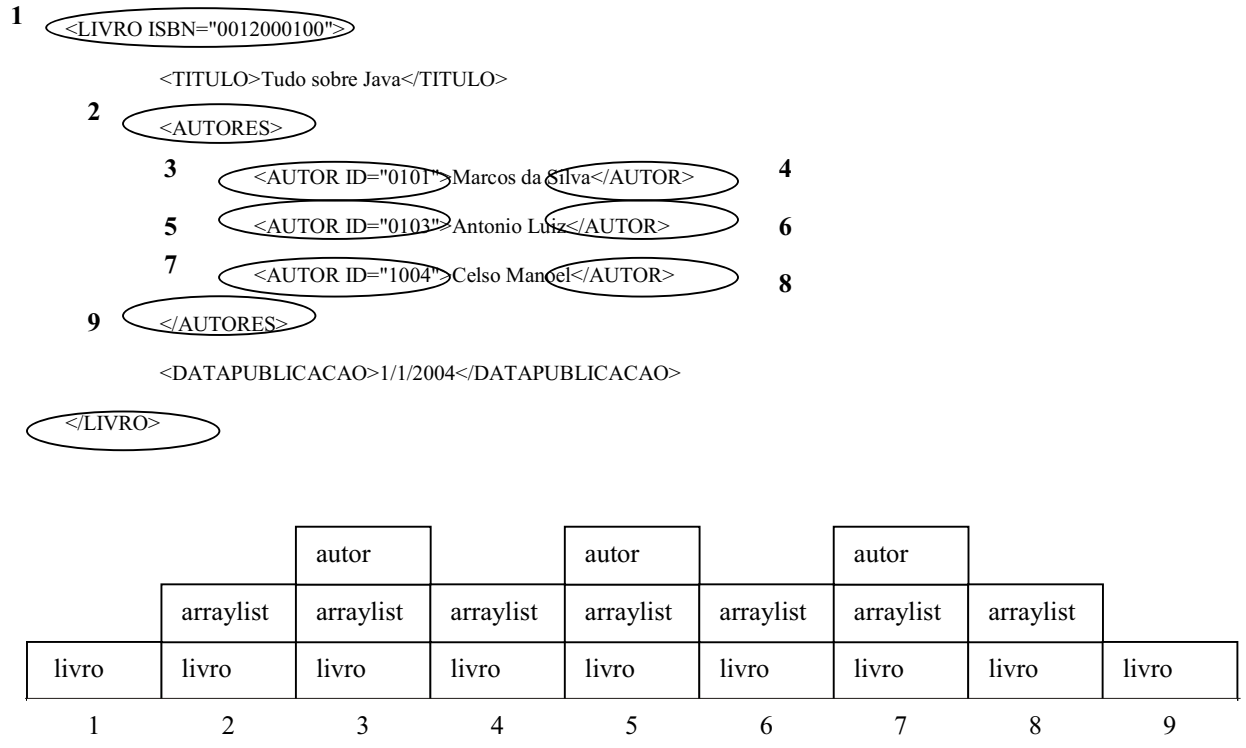

FIG. 5. Passos de processamento do documento XML.

Limitações da Digester

Os exemplos de uso da ferramenta Digester geralmente apresentados (Janert, 2004; Villela, 2003) utilizam suas características básicas. Nestes exemplos os arquivos XML estão organizados de maneira adequada, com valores de *tags* e seus atributos mapeados para variáveis membros de classes Java do tipo *String* (cadeia de caracteres), que acomodam diretamente estes valores. No entanto, nem sempre um atributo do tipo *String* representa adequadamente o tipo de valor que se deseja.

No exemplo acima, o valor "**1/1/2004**", poderia ser melhor traduzido para um objeto da classe *java.util.Date* do Java, o qual permite que sejam diretamente realizadas operações mais coerentes para um dado que representa uma data. Apesar da ferramenta Digester permitir a conversão de valores *String*, correspondente ao valor demarcado por alguma *tag* para o valor da variável membro da classe do tipo *java.util.Date*, percebeu-se que a conversão de valores de atributos de *tags*, para valores de variáveis membro não funciona adequadamente. Uma tentativa de solução utilizando a ferramenta Digester é ilustrada na Fig. 6.

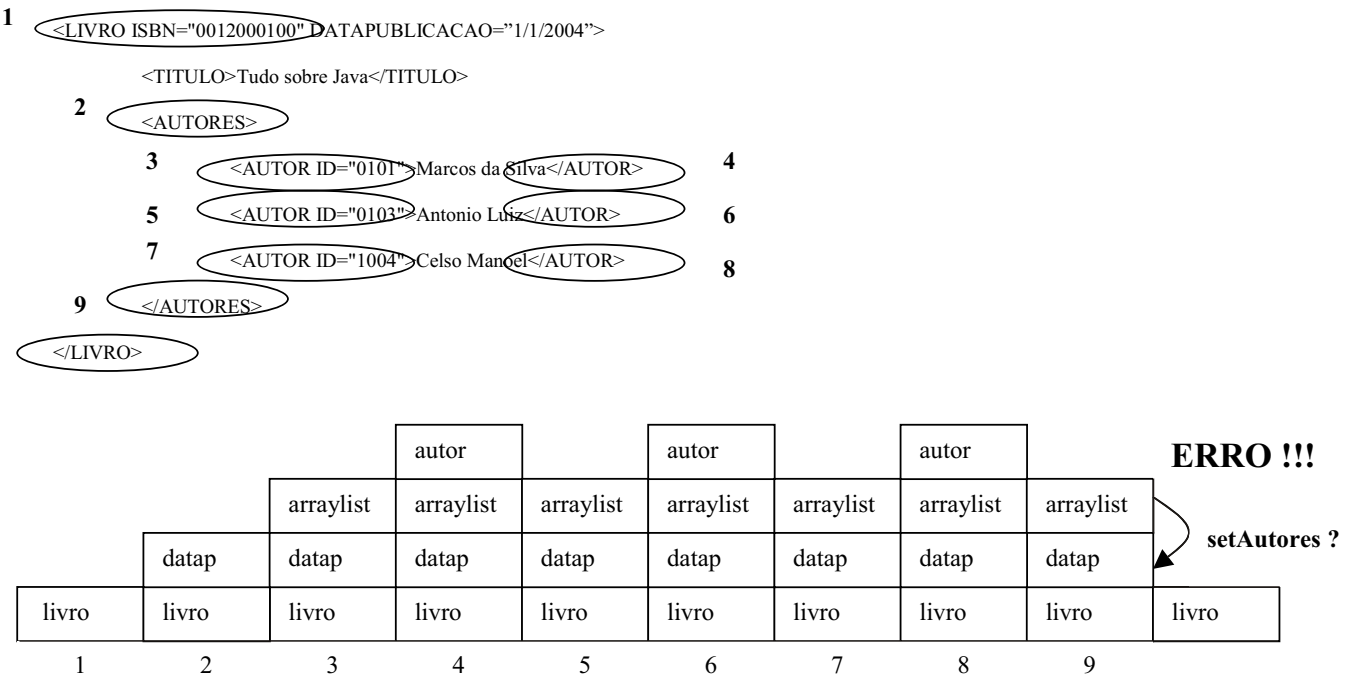

FIG. 6. Mapeamento de atributo XML para objeto do tipo *java.util.Date* com digester.

Nesta figura foi utilizada a regra *ObjectParamRule*, que converte um atributo de uma *tag* XML para um objeto na pilha, que poderá ser então utilizado para atribuir o valor à variável membro do objeto Livro. Neste caso foi encontrado um problema no comportamento da Digester que impede a utilização desta abordagem.

Solução Proposta

Após investigação em seu código fonte observou-se que na Digester, objetos somente são desempilhados depois que a *tag* que originou o empilhamento é fechada, e não quando o objeto é utilizado. Neste contexto, um objeto *datap* (representando a data de publicação), permanece na pilha mesmo após ter sido utilizado para dar valor à variável membro *datapublicacao* de livro. Ao final da análise das regras disponíveis na Digester versão 1.5, não foi encontrada qualquer solução para este problema. Pelos fatores positivos da Digester para a construção de aplicações para processamento de documentos XML decidiu-se então implementar uma nova funcionalidade que suportasse a criação de objetos na pilha a partir de valores de atributos que pudessem ser utilizados sem corrupção da seqüência de empilhamento. Esta funcionalidade está implementada na forma de uma nova regra de nome *FactoryCreateParamRule.*

Durante a análise do código fonte da Digester foi verificada a existência de uma segunda pilha que não é mencionada na documentação. Esta segunda pilha é utilizada no armazenamento de atributos de *tags* para uso posterior por uma regra *CallMethodRule*. Esta pilha é normalmente manipulada quando são encontrados atributos numa *tag* XML, porém sua existência não é amplamente divulgada. Objetos nesta pilha que são utilizados pela regra *CallMethodRule*, são automaticamente eliminados da pilha.

A solução proposta corresponde a uma nova regra implementada na Digester com o nome de *FactoryCreateParamRule.* Esta nova regra utiliza a pilha de atributos de *tags* para armazenamento temporário de objetos. A regra permite que estes objetos sejam criados de acordo com tipos de objeto Java especificados sem interferir na ordem de elementos na pilha de objeto principal da ferramenta e, desta forma, permite maior flexibilidade no tratamento de atributos de *tags*.

Resultados

Um exemplo de uso da nova regra é apresentado na Fig. 7 e o comportamento da pilha é ilustrado na Fig. 8, onde são apresentadas a pilha de objetos normal e a de atributos de *tag*. Usando a nova regra *FactoryCreateParamRule* é possível criar um objeto do tipo *java.util.Date* a partir do valor do atributo DATAPUBLICACAO da *tag* LIVRO, mostrada a seguir:

```
<LIVRO ISBN="0012000100" DATAPUBLICACAO="1/1/2004">
```

Este valor armazenado na pilha de atributos pode ser então atribuído à variável membro do tipo *java.util.Date* da classe Livro, que é um tipo mais adequado para manipulação de datas.

A Fig. 7(c) exemplifica o uso da nova regra no seguinte trecho:

```
...
digester.addCallMethod( "LIVRO", "setDataPublicacao", 1, new Class[]{Date.class});
digester.addFactoryCreateParam( "LIVRO", new CriaDataPublicacao(), 0 );
...
```

O método *addCallMethod* da Digester adiciona uma nova regra *CallMethodRule* para tratamento dos arquivos XML. Esta regra será disparada quando a *tag* LIVRO for finalizada durante o processamento do arquivo. Neste ponto será

executado o método *setDataPublicacao* do objeto no topo da pilha, tendo como argumento um objeto do tipo *java.util.Date.* O valor deste argumento vem da pilha de atributos que deve ter sido previamente preenchida por meio da regra *FactoryCreateParamRule*.

```
<LIVRO ISBN="0012000100" DATAPUBLICACAO=”1/1/2004”>
        <TITULO>Tudo sobre Java</TITULO>
        <AUTORES>
                <AUTOR ID="0101">Marcos da Silva</AUTOR>
                <AUTOR ID="0103">Antonio Luiz</AUTOR>
                <AUTOR ID="1004">Celso Manoel</AUTOR>
        </AUTORES>
</LIVRO>
```

(a)

```
public class Livro {
        private String isbn;
        private String titulo;
        private Collection autores;
        private Date datapublicacao;

        //set´s e get´s para os atributos da classe
        public String toString() { ... }
}

public class Autor {
        private String id;
        private String nome;
        //set´s e gets para os atributos da classe
}
```

(b)

```
import org.apache.commons.digester.Digester;
import org.apache.commons.digester.AbstractObjectCreationFactory;
import org.xml.sax.SAXException;
import java.util.ArrayList;
import java.util.Date;
import java.io.File;
import java.io.IOException;

public class LivroDigester {

   public static void main(String [] args) throws IOException, SAXException
   {
      Digester digester = new Digester();

      digester.addObjectCreate("LIVRO", Livro.class);
      digester.addSetProperties("LIVRO","isbn","isbn");
      digester.addCallMethod( "LIVRO", "setDataPublicacao", 1, new Class[]{Date.class} );
      digester.addFactoryCreateParam( "LIVRO", new CriaDataPublicacao(), 0 );
      digester.addBeanPropertySetter("LIVRO/TITULO","titulo");
      digester.addObjectCreate("LIVRO/AUTORES", ArrayList.class);
      digester.addObjectCreate("LIVRO/AUTORES/AUTOR",Autor.class);
      digester.addSetNext("LIVRO/AUTORES/AUTOR","add");
      digester.addSetProperties("LIVRO/AUTORES/AUTOR","ID","id");
      digester.addBeanPropertySetter("LIVRO/AUTORES/AUTOR","nome");
      digester.addSetNext("LIVRO/AUTORES","setAutores");

      Livro livro = (Livro) digester.parse(new File("livro.xml"));
      System.out.println(livro);
   }

   private static class CriaDataPublicacao extends AbstractObjectCreationFactory
   {
      public Object createObject( org.xml.sax.Attributes attributes )
      {
         String data = attributes.getValue( "DATAPUBLICACAO" );
         Date d = new Date();
         try {   d = new Date(data); }
         catch(Exception e) { };

         return d;
      }
   }
}
```

(c)

FIG. 7. Exemplo de uso de nova regra do Digester (a) Documento em formato XML; (b) Código Java simplificado; (c) Código Java para processamento do documento XML.

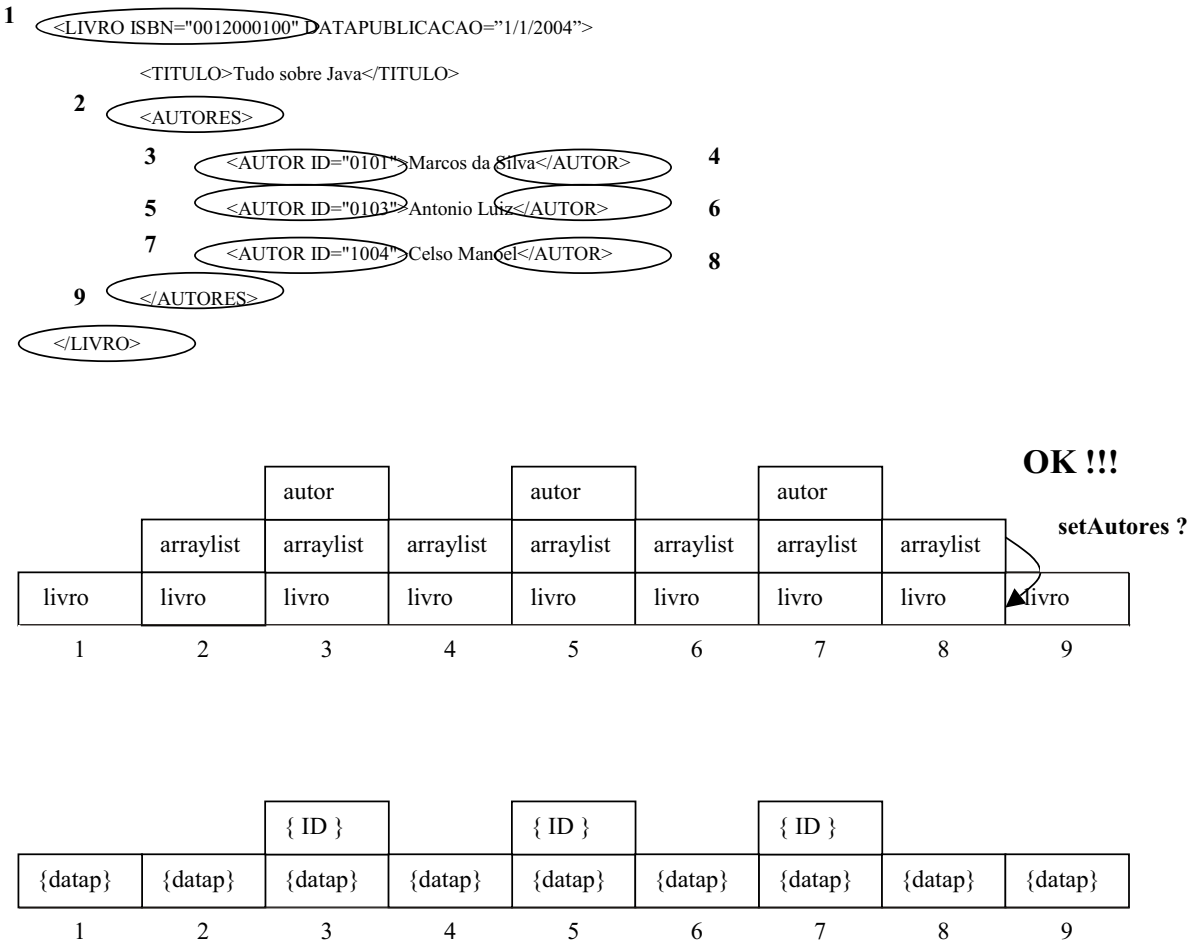

FIG. 8. Comportamento da pilha após inclusão de regra *FactoryCreateParamRule.*

Com o uso desta nova regra está sendo realizado um tratamento mais adequado dos dados existentes nos documentos XML necessários à Agência, permitindo um mapeamento coerente de tipos de variáveis para as classes Java.

Considerações Finais

Ferramentas tais como a Digester, geradas por projetos *open-source*, podem ser utilizadas diretamente no desenvolvimento de aplicações, dando solução imediata para problemas genéricos, que no caso apresentado neste documento corresponde ao processamento de documentos XML. Porém este uso direto pode, algumas vezes, implicar em modificações na aplicação sendo implementada. No caso do Projeto Agência, onde é utilizada a linguagem de programação Java no desenvolvimento de aplicações, estas modificações implicam no surgimento de métodos desnecessários nas classes implementadas. Para que a ferramenta Digester pudesse ser utilizada diretamente no projeto, como ilustrado em algumas publicações, seria necessária a criação de métodos que permitissem a atribuição de valores do tipo String às variáveis membro das classes da aplicação com tipos diferentes de String. Estes métodos, a princípio, não seriam necessários, pois não fazem parte do projeto original das classes e contribuem para torná-las mais complexas.

Outra solução que permite a utilização da versão original da Digester seria a adoção de uma nova estrutura dos documentos XML no projeto Agência. Porém, para verificação do impacto destas mudanças, é necessária uma revisão de todo o código onde os documentos XML são utilizados. Este tipo de modificação pode ser inviável em sistemas legados onde as estruturas dos documentos XML já estão definidas.

Uma outra alternativa é o pré-processamento dos documentos XML utilizando ferramentas de transformação de XML, tal como a linguagem XSLT (World Wide Web Consortium, 2004c), que convertesse documentos XML do projeto Agência para a estrutura processável pela Digester original. Esta alternativa implica na implementação de vários novos *scripts* para transformação de documentos XML e, em um maior custo computacional para conversão.

A solução proposta neste documento tenta diminuir o impacto da adoção da ferramenta Digester no projeto original de classes da Agência transferindo as modificações necessárias para o próprio código da Digester. Com isto espera-se o aperfeiçoamento da ferramenta, permitindo que outros desenvolvedores possam utilizar esta mesma solução.

Referências Bibliográficas

APACHE SOFTWARE FOUNDATION. **The Apache Jakarta Project**: Jakarta Commons. Disponível em: <http://jakarta.apache.org/commons/index.html>. Acesso em: 2 ago. 2004a.

APACHE SOFTWARE FOUNDATION. **The Apache Jakarta Project**: Java related products. Disponível em: <http://jakarta.apache.org/>. Acesso em: 2 ago. 2004b.

APACHE SOFTWARE FOUNDATION. **The Apache Struts web application framework**. Disponível em: <http://struts.apache.org/>. Acesso em: 2 ago. 2004c.

APACHE SOFTWARE FOUNDATION. **The Jakarta Project**: Commons Digester. Disponível em: <http://jakarta.apache.org/commons/digester/>. Acesso em: 2 ago. 2004d.

JANERT, P. K. **Learning and using Jakarta Digester**. Disponível em: <http://www.onjava.com/pub/a/onjava/2002/10/23/digester.html>. Acesso em: 2 ago. 2004.

KROL, E. **The whole internet**: user's guide & catalog. Sebastopol: O'Reilly, 1993. 376 p.

MORDANI, R.; BOAG, S. **Java API for XML processing (JAXP) especification**: version 1.2 final release. Palo Alto: Sun MicroSystems, 2002. 148 p. Disponível em <http://192.18.97.149/ECom/EComTicketServlet/BEGINAE9FD71B45A6E5B12E27F4046CF998FB/2147483648/641086995/1/393818/393806/641086995/2ts+/westCoastFSEND/7024-jaxp-1.2-fr-spec-oth-JSpec/7024-jaxp-1.2-fr-spec-oth-JSpec:1/jaxp-1_2-fr-spec.pdf>. Acesso em: 28 set. 2004.

SAX. **SAX** [home page]. Disponível em: <http://sax.sourceforge.net/>. Acesso em: 30 jun. 2004.

UNIVERSITY OF VIRGINIA LIBRARY. Electronic Text Center. **TEI guidelines for electronic text encoding and interchange.** Disponível em: <http://etext.virginia.edu/bin/tei-tocs?div=DIV1&id=SG>. Acesso em: 6 jul. 2004.

VILLELA, C. Commons Digester acabando com a azia de ler XML. **Revista MundoJava**, ano, 1, n. 1, p. 15-17, set./out. 2003.

WORLD WIDE WEB CONSORTIUM. **eXtensible Markup Language (XML)**. Disponível em: <http://www.w3.org/XML>. Acesso em: 20 jul. 2004a.

WORLD WIDE WEB CONSORTIUM. **W3C HTML home page**. Disponível em: <http://www.w3c.org/MarkUp/>. Acesso em: 20 jul. 2004b.

WORLD WIDE WEB CONSORTIUM. **XSL Transformations (XSLT)**. Disponível em: <http://www.w3.org/TR/xslt>. Acesso em: 3 nov. 2004c.

Comunicado Técnico, 63

Ministério da Agricultura, Pecuária e Abastecimento

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Endereço: Caixa Postal 6041 - Barão Geraldo
13083-970 - Campinas, SP
Fone: (19) 3789-5743
Fax: (19) 3289-9594
e-mail: sac@cnptia.embrapa.com.br

1ª edição on-line - 2004