Automatic Generation of Printed Representations of Ecuadorian Electronic Invoices through XML Data Binding

Rolando P. Rodriguez-Cruz, Maria S. Avila-Garcia, and Maria F. Hernandez-Luquin
DEM Yuriria, University of Guanajuato, Guanajuato, Mexico
Email: {rp.rodriguezcruz, susana.avila, mf.hernandezluquin}@ugto.mx

***Abstract*—This paper presents an approach for the automatic generation of printed representations of Ecuadorian electronic invoices using XML Data Binding. The proposed solution is based on the process of converting XML documents into Java objects and then into PDF documents, according to parameters established by the Ecuadorian legislation. The results were analyzed with regard to the elapsed time during the generation of a printed representation and to the size of resulting objects. The results, obtained as part of the implementation of the solution in four companies in Ecuador, show improvements not only in the generation time of printed electronic invoices but also in more robust and secure mechanisms for handling electronic vouchers through their representations in XML format.**

***Index Terms*—XML, XSD schema, XML Data Binding, XML Mapping, electronic invoice**

I. Introduction

An electronic invoice (e-invoice) is a digital representation of a traditional paper invoice which contains important data for calculating taxes [1]. Generally, an e-invoice is generated by the services supplier as an XML file that contains tax information of the business transaction parties, which is then shared with the customers (receivers). A supplier may additionally generate other related files in formats such as PDF or EDIFACT [2].

According to [3], in recent years several Latin American countries have begun to promote the use of the e-invoice. For instance, in May 2013 the SRI (Servicio de Rentas Internas) in Ecuador published a resolution of mandatory issuance of e-invoices which has had a direct impact on the tax administration processes. According with the Article 4 of the SRI Resolution, published in 2014 and identified as N °NAC-DGERCGC14-00790, the issuer must provide the receiver the necessary tools for accessing the e-invoices in its printed representation in digital format (RIDE).

Given that the e-invoicing Ecuadorian model has three issuance schemes (contingency, online, and offline), issuers must implement mechanisms to dynamically generate RIDEs from heterogeneous XML e-invoices. Although XML is a standard markup language, the different emission schemes mentioned above oblige issuers to implement flexible methods for converting XML e-invoices to non-XML formats such as PDF, a standard for the accurate display of rich and complex material [4].

Since the importance of XML language has increased, a series of features have been developed around it. For instance, XML Data Binding which converts an XML document to an instance of a Data Model Class [5]. XML Data Binding involves four concepts: class generation, unmarshalling, marshalling and binding schemas, as described in [5].

Unmarshalling consists in converting a format (which can vary from a simple raw byte array to an elaborate XML representation) into an object created from an XML Schema Definition (XSD), called Content Tree [6]. This transformation is very important since the direct manipulation of XML documents, represented as strings or DOM trees, can be tedious and error-prone [7].

This paper reports on an approach based on the unmarshalling process to the automatic RIDEs generation from issued XML e-invoices using XSD documents as a skeleton definition. In this work, a prototype implementation targeted to Java programming language has been developed.

The remainder of the paper is organized as follows: Section II analyses the state of the art in the field of XML transformations and highlights the followed path in the design of the proposed tool. Section III presents the XML Unmarshalling process using block diagrams. Section IV describes a specific implementation based on the Ecuadorian e-invoice model using JAXB (Java Architecture for XML Binding) and the obtained results. Finally, Section V presents conclusions and future work.

II. Related Work

In recent years, there has been a continuous evolution in the field of the correct interpretation and handling of XML documents with heterogeneous schemas. Several approaches have been reported in three related topics: XML generation from XSD schema, XML to data model

Manuscript received February 26, 2016; revised May 21, 2016.
This work was supported by the Mexican "National Council on Science and Technology," CONACYT

mapping, and XML Data Binding Applications, which are discussed in the following subsections.

A. XML from XSD Schema/Marshalling

An XSD schema specifies the content that resides in an XML document using the XML Schema Definition Language [8].

References [9-10] present a technique for dynamically creating an XML document according to a given XSD schema using a computer language to generate a tree linked structure. Fig. 1 shows the process described in these proposals.

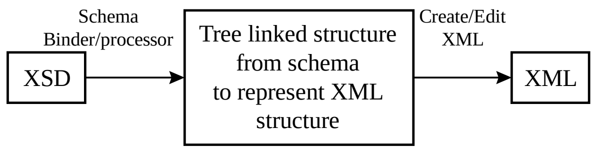

Schema Binder/processor
XSD
Tree linked structure from schema to represent XML structure
Create/Edit XML
XML

Figure 1. Generation process of an XML document from an XSD schema as described in [9-10].

Reference [9] implements a technique using Java DOM API, and the proposal shown in [10] is based on JAXB. Both of them proposed determining the efficiency of their methods as future work.

B. XML to Data Model Mapping

Since the introduction and dissemination of the XML standard, there have been issues associated with handling this data representation format. For example, as we can see in Fig. 2, a single real-world object can be represented in many ways through an XML file [11]. Additional tools are needed to resolve conflicts caused by the heterogeneity of data models.

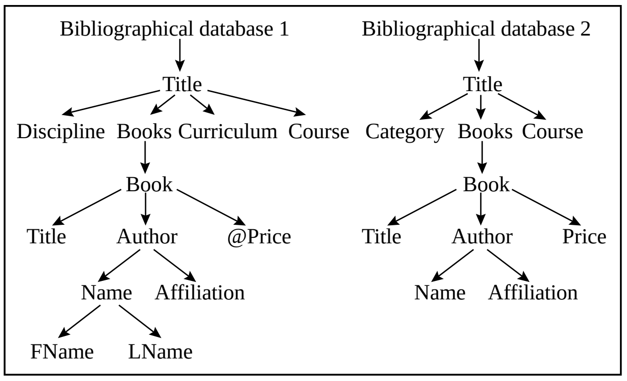


Figure 2. Bibliographical databases of different publishers may be different [11].

Reference [11] reports an approach for resolving structural and semantic conflicts of heterogeneous XSD schemas in order to deal with data of different sources which are represented in different formats and in incompatible ways. This approach was reported as an initial prototype; some issues, such as many-to-many correspondences and integrity constraints, have not yet been sorted out.

A transformation approach for responding to the variation in different styles of XSD schema's organization as needed in XML to object mappings is presented in [12]. A semantic mapping method which finds the correspondence between two XML schemas, even if the schemas have different structures, is introduced in [8].

C. XML Data Binding Applications

Proposals presented in [13]-[16] use information extracted from XML documents to generate mobile applications, widgets, documents or GUI (Graphical User Interfaces). For example, Reference [13] presents an approach to generate compact applications to process XML documents, based on the fact that applications that make use of XML data with large schemas do not necessarily use all of the information included in them.

Reference [14] proposes a translator to represent data defined in XSD schema into graphic user interfaces using Apache Xerces-J. First, the translator generates a DOM tree representation of nodes, then a second module dynamically generates a user interface for mobile devices in different rendering languages such as Java Swings, HTML and WML.

An easily extensible tool to generate automatically functional widgets from XML documents based on the Widget Markup Language using JAXB is proposed in [15].

The authors of [16] present an XML based Meta model to represent UML classes and stored-procedures to access databases which allow high independence between databases and the software components.

Our approach complies with the Electronic invoicing process of Ecuador. It uses the technique proposed in [10] to create an XML document, and JAXB as technology to implement XML data binding because it provides a coherent and standard interface for XML marshalling, unmarshalling, and supports compiling and runtime binding [15], [17].

III. Methodology

Unmarshalling process allows to build object trees from XML documents that are instances of an XSD schema [6]. Each element in the tree corresponds to an element in the XML document. Fig. 3 illustrates the entire Unmarshalling process.

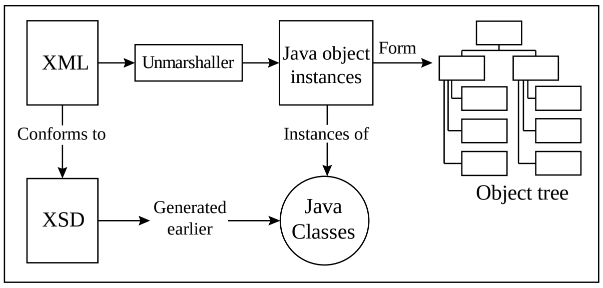


Figure 3. Unmarshalling process flow as described in [5].

The overall Unmarshalling workflow shown in Fig. 3 is described in the following sections:

A. XML to Data Model Mapping

Given an XSD schema, we can easily define the corresponding structure in a computer language using the method reported in [10]. We use JAXB to generate the Java classes as suggested in [10] and [17]. JAXB provides a tool called binding compiler to bind the XML schema. It generates a set of java interfaces and classes that represent the XML schema. The Java classes generated must also define *get* and *set* methods which are used to obtain and specify data for each type of element and attribute in the schema.

B. XML Data Conforms to XSD Schema

XML data is necessary to continue with the Unmarshalling process. The XML document must follow the rules defined in the XSD document [5]; in other words, XML properties and data attributes are restricted by an XSD schema [18].

C. XML Data into Instances of Java Classes

According to [5], in this step XML data is converted into some form of an input stream. Each property of XML document must match with the respective attribute of the Java object instantiated.

Once the Unmarshalling process has been performed, it returns a top-level instance of the XML document. This is an instance of the class that corresponds to the root element of the XML document. The result is a normal and plain Java object with *set* and *get* methods. This object (also called object tree) can now be used as an input to functions entrusted to render the RIDE in formats such as PDF.

D. Object to PDF Content Matching

As a result of the previous steps, we have now a cached copy of the content from the XML source document in a Java object. The next stage of the RIDE generation process is to structure the object's attributes inside of the PDF.

Although the order used to represent the structure of an e-invoice in a PDF file is not the same as the one used by XML or an object instance, we can use a relatively simple algorithm to search for the position of each XML field in the PDF content. First, the document type of electronic invoice is determined in order to define the data that must be rendered through a PDF file. Once this happens, each attribute of the object tree will be retrieved via encapsulation using get methods (See Fig. 4). Once the algorithm has finished, a PDF file is created.

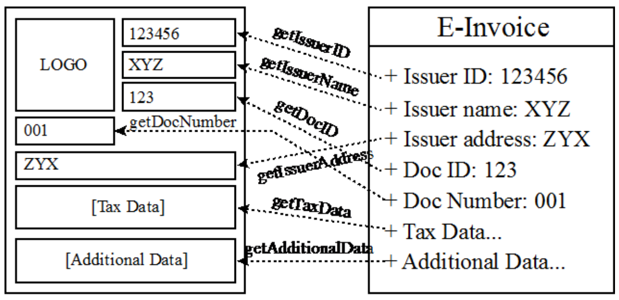


Figure 4. Each attribute of the object tree is recovered through its get method, then it is placed in its corresponding position within the PDF content.

IV. Implementation

According to the Ecuadorian legislation, taxpayers must issue the following electronic documents:

- Invoices
- Bills of lading
- Credit notes
- Debit notes
- Tax withholding receipts

which are supported by the developed application. The SRI offers in its website [19], XSD documents for each type of electronic documents.

An example of a XSD definition is shown in Fig. 5.

```
<xsd:element name="factura">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name="infoTributaria"
                          type="infoTributaria"/>
    <xsd:element name="infoFactura">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="fechaEmision"
        type="fechaEmision"/>
       <xsd:element name="dirEstablecimiento"
        type="dirEstablecimiento"/>
       <xsd:element name="contribuyenteEspecial"
        type="contribuyenteEspecial"/>
       <xsd:element name="obligadoContabilidad"
        type="obligadoContabilidad"/>
       <xsd:element name="guiaRemision"
        type="guiaRemision" minOccurs="0"/>
       <xsd:element name="razonSocialComprador"
        type="razonSocialComprador"/>
```

Figure 5. Partial definition of an electronic invoice through the XSD schema.

All the needed classes associated to each type of electronic document were obtained using JAXB. An example of the translation of the XSD schema of an e-invoice into a Java class is shown in Fig. 6.

```
@XmlRootElement(name = "factura")
public class Factura {
  @XmlElement(required = true)
  protected InfoTributaria infoTributaria;
  @XmlElement(required = true)
  protected InfoFactura infoFactura;
  @XmlElement(required = true)
  protected Detalles detalles;
  protected InfoAdicional infoAdicional;
  @XmlAttribute protected String id;
  @XmlAttribute
  @XmlSchemaType(name = "anySimpleType")
  protected String version;
  ...
}
```

Figure 6. Translation of the XSD schema of an e-invoice in Java using JAXB.

Using the Java class shown in Fig. 6, we can instance a Java object tree from the XML document shown in Fig. 7, that are instances of the schema used to generate the Java classes.

```
<?xml version="1.0" encoding="UTF-8"?>
<factura
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 id="comprobante" version="1.0.0">
 <infoTributaria>
  <ambiente>1</ambiente>
  <tipoEmision>1</tipoEmision>
  <razonSocial>
   PRODUCTOS ECUATORIANOS S.A.
  </razonSocial>
  <nombreComercial>
   PRODUCTOS ECUATORIANOS SA
  </nombreComercial>
  <ruc>1190381595001</ruc>
  <claveAcceso>
   0611201501119038159500110040030014554370145
  </claveAcceso>
  <codDoc>01</codDoc>
  <estab>004</estab>
  <ptoEmi>003</ptoEmi>
  <secuencial>001455437</secuencial>
  <dirMatriz>
   TAMAYO 1025 Edif CLASECUADOR
  </dirMatriz>
 </infoTributaria>
```

Figure 7. Partial XML example of an e-invoice.

The Java object shown in Fig. 8 will allows the generation of the RIDE as it contains all information of the electronic invoice.

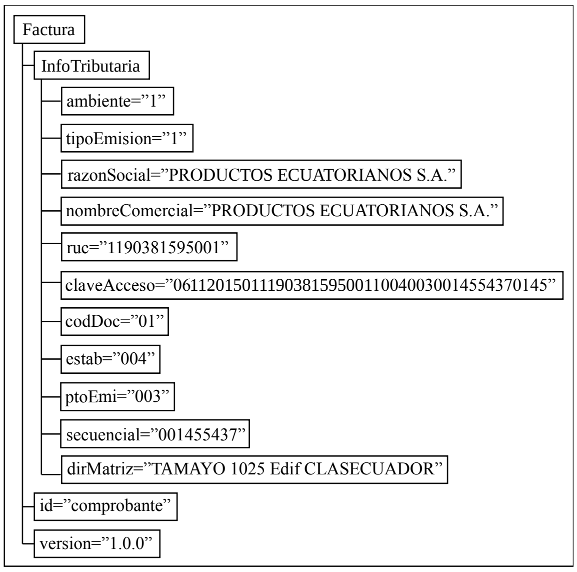

Factura
InfoTributaria
ambiente=”1”
tipoEmision=”1”
razonSocial=”PRODUCTOS ECUATORIANOS S.A.”
nombreComercial=”PRODUCTOS ECUATORIANOS S.A.”
ruc=”1190381595001”
claveAcceso=”0611201501119038159500110040030014554370145”
codDoc=”01”
estab=”004”
ptoEmi=”003”
secuencial=”001455437”
dirMatriz=”TAMAYO 1025 Edif CLASECUADOR”
id=”comprobante”
version=”1.0.0”

Figure 8. Object instance tree for an e-invoice.

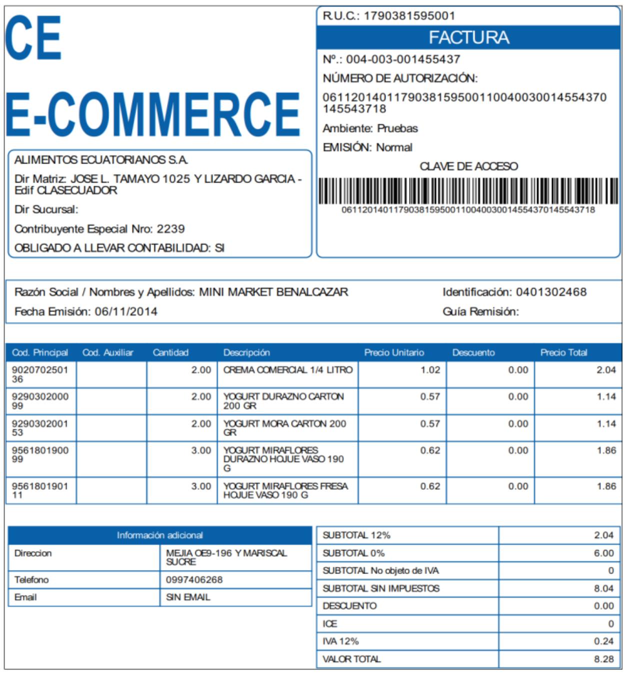
The next stage of the generation process is to translate the Java object containing the information of an e-invoice and to build its corresponding PDF document. Fig. 9 presents the printable version of an e-invoice generated from the object of the Fig. 8 through the Unmarshalling process.

The RIDE’s generator proposed in this work can be deployed:

- In standalone mode, where we can use the application as an external jar library in any Java project, and
- As a web service, where the application can be used through SOAP protocol, ensuring extensibility, transparency and interoperability [19].

The iText rendering engine, a library that allows PDF file manipulation, was used achieving good results.

The proposed software tool developed for this work is on production stage in four companies in Ecuador. Before the deployment of the proposed tool, the RIDEs were generated using string processing algorithms, which are tedious and unsafe [7]. The generator has made an impact in the working practices of these companies reducing the time needed to generate the RIDE’s. Furthermore, our approach provides a more robust and secure mechanism for handling the data of the e-invoices through their representations in XML format.


CE

E-COMMERCE

R.U.C.: 1790381595001

FACTURA

Nº.: 004-003-001455437

NÚMERO DE AUTORIZACIÓN:

0611201401179038159500110040030014554370145543718

Ambiente: Pruebas

EMISIÓN: Normal

CLAVE DE ACCESO

0611201401179038159500110040030014554370145543718

ALIMENTOS ECUATORIANOS S.A.

Dir Matriz: JOSE L. TAMAYO 1025 Y LIZARDO GARCIA - Edif CLASECUADOR

Dir Sucursal:

Contribuyente Especial Nro: 2239

OBLIGADO A LLEVAR CONTABILIDAD: SI

Razón Social / Nombres y Apellidos: MINI MARKET BENALCAZAR — Identificación: 0401302468

Fecha Emisión: 06/11/2014 — Guía Remisión:

Cod. Principal	Cod. Auxiliar	Cantidad	Descripción	Precio Unitario	Descuento	Precio Total
902070250136		2.00	CREMA COMERCIAL 1/4 LITRO	1.02	0.00	2.04
929030200099		2.00	YOGURT DURAZNO CARTON 200 GR	0.57	0.00	1.14
929030200153		2.00	YOGURT MORA CARTON 200 GR	0.57	0.00	1.14
956180190099		3.00	YOGURT MIRAFLORES DURAZNO HOJUE VASO 190 G	0.62	0.00	1.86
956180190111		3.00	YOGURT MIRAFLORES FRESA HOJUE VASO 190 G	0.62	0.00	1.86

Información adicional	
Direccion	MEJIA OE9-196 Y MARISCAL SUCRE
Telefono	0997406268
Email	SIN EMAIL

SUBTOTAL 12%	2.04
SUBTOTAL 0%	6.00
SUBTOTAL No objeto de IVA	0
SUBTOTAL SIN IMPUESTOS	8.04
DESCUENTO	0.00
ICE	0
IVA 12%	0.24
VALOR TOTAL	8.28

Figure 9. Printable version of an e-invoice in PDF format generated from a Java object using iText.

V. Conclusions and Future Works

This paper has presented our approach to the development of a tool to generate automatically RIDEs from e-invoices according to Ecuadorian model using Unmarshalling. The developed tool is easily extensible to future technologies, as well as to new issuance schemes. Furthermore, the generator can be exploited as a standalone application or as a library in the context of other applications.

The main contributions of our approach can be summarized as following:

- Enterprises no longer need to handle complicated e-invoicing processes to generate RIDEs. The developed tool determines the issuance scheme of e-invoices automatically.
- Through automated generation of RIDEs with a robust process, e-invoicing frameworks will be able to deal with the dynamic changes in polices in the e-invoicing model of Ecuador.
- The manipulation of e-invoices through objects or data structures is safer than the direct manipulation of XML files [7], because objects provide a high-

level specification. Also, objects represent a best option for Non-XML databases.

Future work is focused in the extension of the proposed tool for cloud and mobile technologies. Moreover, we intend to enhance the generator with mechanisms proposed in [20], in order to improve the semantic structure of the PDF files according to the logical structure of its corresponding XML file.

ACKNOWLEDGMENT

The authors wish to thank to the Mexican "National Council on Science and Technology," CONACYT, for the financial support provided via the scholarships 444186 and 664689 granted to Rolando P. Rodriguez-Cruz and Maria F. Hernandez-Luquin respectively.

REFERENCES

[1] L. Humski, I. Lazegic, and Z. Skocir, "Datawarehouse for fer e-invoice system," in *Proc. 35th International Convention MIPRO*, May 2012, pp. 1641–1646.

[2] European Commission, Definitions for e-Invoicing, 2012.

[3] Billentis, "E-invoicing/e-billing: Key stakeholders as game changers," 2014.

[4] M. R. B. Hardy and D. F. Brailsford, "Mapping and displaying structural transformations between xml and pdf," in *Proc. 2002 ACM Symposium on Document Engineering*, DocEng '02, USA: New York, ACM, 2002, pp. 95–102.

[5] B. McLaughlin, *Java and XML Data Binding*, O'Reilly, ch. 1, 2002, pp. 16-18.

[6] Sun Microsystems, Inc., *The Java Architecture for XML Binding Users Guide*, ch. 3, 2001, pp 24-26.

[7] C. Kirkegaard, A. Moller, and M. I. Schwartzbach, "Static analysis of xml transformations in java," *IEEE Transactions on Software Engineering*, vol. 30, no. 3, 2004, pp. 181–192.

[8] L. Checiu and D. Ionescu, "A new algorithm for mapping xml schema to xml schema," in *Proc. 2010 International Joint Conference on Computational Cybernetics and Technical Informatics*, 2010, pp. 625–630.

[9] P. Pardeshi and G. Ramachandran, "Generic method for xml generation from given xsd," in *Proc. 3rd International Conference on Electronics Computer Technology*, vol. 5, pp. 404–408, April 2011.

[10] D. Raha, "Dynamic xml generation according to a given schema," in *Proc. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1, Dec. 2008, pp. 938–942.

[11] A. Almarimi and J. Pokorny, "A mediation layer for heterogeneous xml schemas," *International Journal of Web Information Systems*, vol. 1, no. 1, pp. 25–33, 2005.

[12] R. Lammel, "Style normalization for canonical x-to-o mappings," in *Proc. 2007 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation*, 2007, pp. 31–40.

[13] A. Tamayo, C. Granell, and J. Huerta, "Instance-based xml data binding for mobile devices," in *Proc. the Third International Workshop on Middleware for Pervasive Mobile and Embedded Computing*, MMPAC '11, USA: New York, ACM, 2011, pp. 2:1–2:8.

[14] V. Radha, S. Ramakrishna, and N. P. kumar, "Generic xml schema definition (xsd) to gui translator," in *Proc. the Second International Conference on Distributed Computing and Internet Technology*, Berlin, Heidelberg: Springer-Verlag, 2005, pp. 290-296.

[15] C. Raibulet and D. Cammareri, "Automatic generation of mobile widgets," *International Journal of Pervasive Computing and Communications*, vol. 7, no. 2, pp. 132–146, 2011.

[16] J. Gunathunga, A. Umagiliya, and S. Kodituwakku, "Leverage the use of xml in dynamic gui parsing and database stored procedures," in *Proc. International Conference on Industrial and Information Systems*, Aug 2007, pp. 235–238.

[17] T. Aihkisalo and T. Paaso, "A performance comparison of web service object marshalling and unmarshalling solutions," in *Proc. 2011 IEEE World Congress on Services*, USA: Washington, DC, IEEE Computer Society, 2011, pp. 122–129.

[18] K. Svensson, "Faster xml data validation in a programming language with xml datatypes," *SIGPLAN Not.*, vol. 42, Nov. 2007, pp. 15–21.

[19] J. Tekli, E. Damiani, R. Chbeir, and G. Gianini, "Soap processing performance and enhancement," *IEEE Transactions on Services Computing*, vol. 5, Third 2012, pp. 387–403.

[20] M. R. B. Hardy, D. F. Brailsford, and P. L. Thomas, "Creating structured pdf files using xml templates," in *Proc. 2004 ACM Symposium on Document Engineering, DocEng '04*, USA: New York, ACM, 2004, pp. 99–108.

Rolando P. Rodriguez-Cruz was born in Trujillo province, Peru, 1988. He received his Bachelor Degree in Computer Science from National University of Trujillo, Peru in 2011 and he is currently pursuing master's degrees in Administration of Technologies at University of Guanajuato, Mexico. He has got a total of 4 years as consultant on electronic invoicing for companies of Ecuador. He is currently Professor at Engineering Division, University of Guanajuato, Guanajuato, Mexico. His area of interest include electronic business, software project management, software process improvement and web technologies.

Maria S. Avila-Garcia is an associate professor at the University of Guanajuato. Her research interests lie in multidisciplinary projects, research information systems, collaborative learning and research environments, and medical image processing.

Maria F. Hernandez-Luquin received her Bachelor Degree in Computing Engineering from Engineering Division of the University of Guanajuato, Mexico in 2014. Now, she is pursuing master's degrees in Applied Electronic Engineering at University of Guanajuato, Mexico. She is currently Professor at Engineering Division, University of Guanajuato, Guanajuato, Mexico. Her current research interests include software development and distributed databases, pattern recognition and image processing.