Integrity for form-based input: Towards an XML Schema based approach

Author and co-authors

John van Zyl

Reinhardt A. Botha

Dalenca Pottas

Author's affiliation

Faculty of Computer Studies

Port Elizabeth Technikon

Author's contact details

{jvanzyl, reinhard, dalenca}@petech.ac.za

041-5043313

Faculty of Computer Studies

Port Elizabeth Technikon

Private Bag X6011

Port Elizabeth, 6000

Integrity for form-based input: Towards an XML Schema based approach

Abstract

The eXtensible Markup Language (XML) as identified in the XML 1.0 W3C Recommendation, this year (2003) turned five. One of the reasons for its popularity is the fact that XML inherently provides a degree of integrity. Not only is it a formal language, which allows no exceptions, but it is also a language that provides for further integrity checking against XML schemas. An XML Schema contains important structural (and thus integrity) information regarding the document it describes. One of the other factors for the success of XML is its simple text nature. Having the XML documents available as text, has several advantages. Not only is it easy to access programmatically, but it can be read by humans as well. This however can be a disadvantage for human users, who would not want to capture new XML documents as pure text. For ease of use users would obviously prefer to have alternative ways of capturing information. One such way would be to utilize form-based input approaches. This paper discusses the use of XML schemas to produce such form-based input applications. The proposed approach utilizes some of the meta data supplied as part of the XML Schema to produce form-based input applications that preserve the inherent integrity requirements. In particular, the paper will comment on the implementation of such forms as HTML forms. However, in general, the meta data extracted from the XML schema would allow us to generate a wide variety of intermediate source code to handle other form-based input mechanisms such as the candidate XForms recommendation or Windows forms.

Keywords:

Semantic Integrity, XML Schema, Form-based Input, Usability

Integrity for form-based input: Towards an XML Schema based approach

1 Introduction

Extensible Markup Language (XML), as identified in the XML 1.0 W3C Recommendation, this year will have been around for five years. The staying power of XML is primarily due to the fact that it is simple (plain text) and it uses self-descriptive text (markup tags).

XML is a foundation technology that can be extended by various adjunct technologies. Some examples of such technologies are, XSLT, XPath and Xquery (Dick, 2000). With this sound foundation XML can be used to structure, describe, and interchange any form of data imaginable. An intended effect of these features is that XML is capable of storing data. Another important factor that distinguishes XML from other markup languages is the level of integrity that this technology has to adhere to.

XML is not only a formal language, tolerating no exceptions, but also a language that can be further checked for validity. In short XML documents must be well formed and it must be valid. On the one hand, for XML documents to be well formed, one of the aspects it has to adhere to is that for example a start tag must be accompanied by an associative end tag (Bray, Paoli, Sperberg-Mcqueen, & Maler, 2000). On the other hand to ensure validity of XML documents, these documents, must conform to accompanied Document Type Definitions (DTD) or to XML Schema specifications. A DTD is a file containing important integrity information about the XML document it is describing. A similar effect is achieved through the use of the XML schema specification. However a schema specification provides greater means of describing a document than that provided by DTDs. This information then ensures a high level of integrity and is used to concisely produce and manipulate these type of documents (M:XML.com(Walsh):98:XMLIntroduction, 1998).

Seeing that XML documents are only text based, although an advantage, for simple reasons, it also has a negative affect on usability. An example of this can be found in a typical content based environment where users on different levels are required to perform various forms of inputs and data capturing (Gropp, 2003). These users would not want to perform tedious tasks on text-based interfaces. For ease of use users surely would prefer to have some form-based approach to perform these tasks.

It is then a logical thought that for XML to reform from a text based view approach to a more user friendly view, a model must be devised to produce usable form-based input forms. Some of the characteristics of such a model is that it needs to produce an approach that adheres to a high level of integrity (technical) aspects but also adheres to a high level of usability integrity (known good practise guidelines) features. These characteristics subsequently manipulate human aspects that in turn complement the level of correctness of data being captured. This leads to an added advantage that a higher level of integrity of the resultant XML document output is achieved.

From the preceding discussion, it is clear that integrity aspects of resultant XML document outputs have a principal role to play in the success of an approach to form-based input. To this effect, the next sections take a closer look at XML as a standard for information representation with specific reference to the integrity of information. From this the various types of information integrity are determined and subsequently discussed. This leads to determining how XML conforms to this integrity. A further aspect that is addressed is the matter of usability, with the focus on identifying what usability features form-based input requires.

The paper culminates in the discussion of a model for the conversion of text-based input to form-based input applications that preserve the inherent integrity requirements and address particular usability features.

2 XML: A Standard for Information Representation

The Extensible Markup Language, like HTML, is a markup language that makes use of tags. The main advantage in the use of XML markup is the fact that in XML, descriptive markup tags are used to describe and focus on the content and structure of the document. These descriptive markup tags encapsulate the text and describe what the text is.

The main reason why XML is capable to do this is the fact that XML is a formal language. As the case is with any formal language, it has to adhere to various grammar rules. The successful use of such grammar rules then allows the description of content that can be stored within XML documents in a well formed structured manner (Bray et al., 2000).

XML does not only allow the description of its content, but it can also be used to describe the rendering of this content. This can be achieved by applying different stylesheets to the same document. Therefore, the information resides in the XML document, while the rendering information (stylesheet) is elsewhere. This demonstrates that in an XML document the presentation of the document is separate from the content (Dick, 2000).

The fact that XML distinguishes between content and presentation, allows the description of information. It should be realized that the XML standard places great emphasis on content.

Information is a critical asset in any organization. The use of HTML and related HTML technologies, for example the Internet, Intranet and Extranets, made it possible for information to be made available across such networks. With the advent of XML which allows similar accessibility, the added bonus is that it also allows the representation of information in a structured manner.

More and more organizations are conforming to this standard to allow this accessibility to and description of structured data to various users on corporate networks. An important issue that arises with this conformance to the XML standard, is an effective and efficient method to preserve integrity of the information that is being represented by various XML documents (M:XML.com(Walsh):98:XMLIntroduction, 1998).

To understand the preservation of integrity, the various information integrity features that are required need to be determined, as discussed in the following section.

3 Integrity

Information integrity points to a property of information, i.e. that information has an unimpaired soundness. Information integrity is a concern whenever information is stored or in transit. When considering information that is in transit, discussions on information integrity generally relate to: the integrity of a single data unit or field and the integrity of a stream of data units or fields (Leyman & Roller, 2000).

The integrity of a single data unit or field on the one hand, is concerned with the preservation of data against modification-the data must be valid. The integrity of a stream of data units or fields, on the other hand, is the preservation of data against misordering, loss or inserting. In other words: the data must be complete.

Various approaches and techniques are employed to ensure that information stays valid and complete: encipherment, authentication, time stamps, information digest, access control, and so forth (Donatello, 1989). When data is in transit, information integrity thus refers to two complementary components: validity, and completeness.

When information is stored (structured), typically on a database, but now also on XML documents, the validity and completeness of the information is also important. To be certain of the integrity of stored information users need assurance that (Motro, 1998):

- all invalid/false information is excluded from the database, (i.e. only valid information is stored);

- all correct information is included in the database (i.e. the information is complete).

It can thus be argued that the integrity of information, both in transit and in storage, is critical to any organization.

The unimpaired soundness of stored information (relational databases and XML documents) is vital to ensure the integrity of an organization's critical information. To ensure the integrity of information, three types of integrity measurements can be observed (Leyman & Roller, 2000):

- operational integrity measures: integrity that deals with the synchronization of concurrent access to data.
- physical integrity measures: integrity that protects from the loss of data due to media failures.
- semantic integrity measures: integrity that ensures that data complies to the appropriate business rules.

Of these, semantic integrity is the only type that allows for the reflection of the day-to-day reality in which businesses perform commerce. The reality of day-to-day business operations is modelled through various business processes and business rules. The main objectives of these rules are to control and successfully implement the various business processes that exist within an organization. Business processes are typically depicted and supported through the use of information systems and more specifically the use of various data sources.

It can be stated that semantic integrity forms the core for ensuring integrity of information that is typically stored within relational databases and on XML documents.

4 Semantic Integrity

As what was seen in the previous section semantic integrity allows the operational implementation of ever changing business rules within an organization. The following sections discusses the occurrence of semantic integrity within databases as well as XML approaches.

4.1 Semantic Integrity and Relational Databases

The occurrence of information integrity within relational databases is obtained by means of semantic integrity constraints. Integrity constraints are formulas in predicate calculus that express relationships that must be satisfied by the database (North, 2000). Thus, in assuming that initially a database satisfies the constraints that were enforced, update requests occurring thereafter are only accepted if these requests do not violate any of the constraints that were enforced (Motro, 1998). These constraints are then typically modelled on actual business rules, determined within the business. Examples of such constraints are PRIMARY KEY, DATATYPES, NOT NULL, DEFAULT, CHECK and TRIGGERS.

Figure 1 depicts a business with its various activities. It shows how these business activities are represented in information systems using databases. The figure further continues to show how the embodiment of policies and strategies results in business rules that govern the way a business goes about performing their daily processes. These business rules are then shown to determine how relational database integrity constraints are implemented.

4.2 Semantic Integrity in XML

XML, as is the case with relational databases, is capable of storing structured content. The conformance to semantic integrity in XML leads to a brief discussion on DTDs and XML schemas.

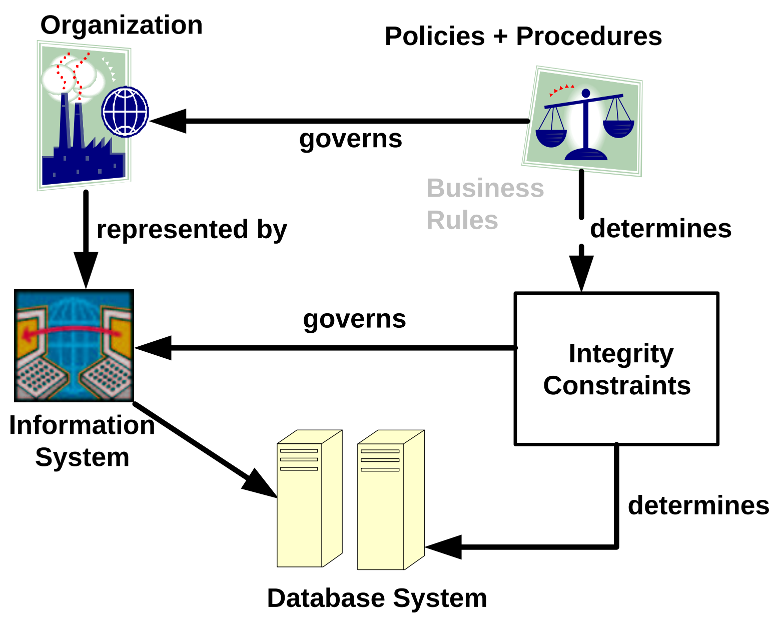

Figure 1: Semantic Integrity and Business.

4.2.1 DTD

In essence a DTD is a Document Type Definition Language, which means that it is a grammar used to restrict tags and the structure of an XML document. This language then defines the rules tags must comply to, to ensure the integrity of the information being represented by these tags.

A DTD is actually a file that is included externally or internally with an XML document. The file consists of the formal definitions as provided by the definition language. These definitions describesw rules that XML documents have to conform to, for it to be valid (Damiani, Vimercati, Paraboschi, & Samarati, 2000).

However DTDs currently only provide limited support for defining what would be recognized as database schemas or object data types (Manola, 1999). The biggest limitation DTDs have, is the fact that DTDs do not support namespaces. This together with the fact that its data types are weak and only apply to attributes, are some of the main motivations for the need to develop new schema languages.

4.2.2 XML Schemas

XML schemas are similar to DTDs but provide functionality beyond those provided by this definition language. XML schemas express shared vocabularies and like DTD's allow machines to carry out additional rules made by people. They provide a means for defining the structure and semantics for XML documents. This is made possible through the use of specifications as agreed on by the World Wide Consortium. These specifications provide an environment where rules can be declared for the accompanied XML document that are very similar to the rules provided by relational database schemas.

The XML Schema specifications provided by the W3C consortium consist of various parts that are grouped into a non-normative specification and a normative specification. The normative specification parts are: XML Schema Part 1 and XML Schema Part 2. They provide specifications to be used with XML datatypes and structures respectively. The non-normative document, XML Schema Part 0, is a primer that uses some of the general, more frequently used specifications as contained in the normative documents. To make use of these specifications it is required to include a namespace reference in the XML Schema document, to enable the use of the respective specifications the namespace represents.

An example of the "XML Schema Part 0: Primer specification" declarations can be found in the use of the enumeration facet. The enumeration facet allows the declaration of user defined types. These types are derived from existing built in simple types and involves the placing of a restriction on them (Bray et al., 2000). These user defined types allow the declaration of an element against a simple type that is customized according to the needs of a specific business environment.

The following code segment depicts such an implementation.

```
<xsd:simpleType name="currencies">
   <xsd:restriction base="xsd:string">
       <xsd:enumeration value="DKR" />
       <xsd:enumeration value="EUR"
       <xsd:enumeration value="GBP" />
```

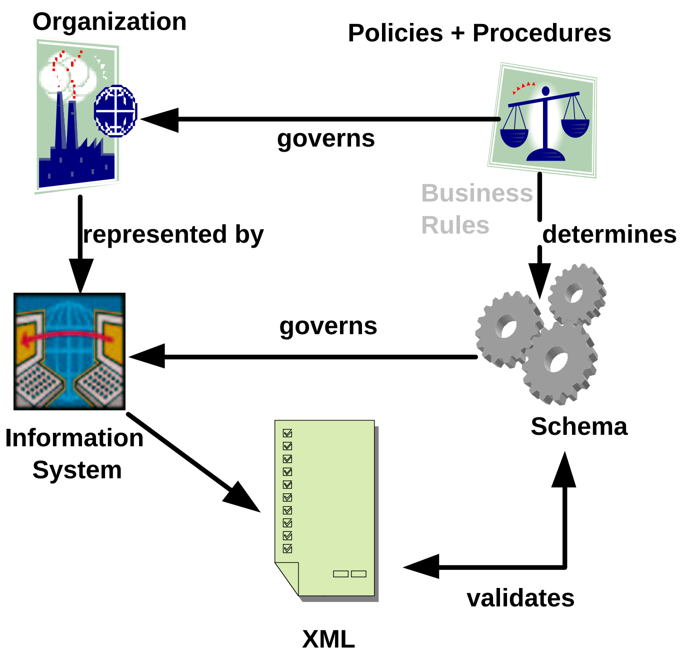

Figure 2: Semantic Integrity and XML.

```
            <xsd:enumeration value="SEK" />
            <xsd:enumeration value="USD" />
            <xsd:enumeration value="NOK" />
        </xsd:restriction>
    </xsd:simpleType>
```

The code segment depicts a declaration for valid currencies. The only entries that will be validated successfully are those entries contained in the various enumeration declarations, i.e. DKR, EUR, GBP and so forth.

The use of these specifications for example enumeration, allows the achievement of the appropriate structures that would ensure semantic integrity of data that is being represented in XML documents. The result is akin to what is achieved by means of using integrity constraints in relational databases. To describe XML and semantic integrity, a very similar description is provided to what was explained in Figure 1. Figure 2 depicts these similarities.

In Figure 1 it is clear that in the traditional database driven approach, the various business activities are represented through the use of information systems. In Figure 2 these activities now are enabled for use in various environments through the use of XML documents. These XML documents again are validated by appropriate schemas. These schemas are again derived from the various business rules that an organization adheres to. A similar integrity effect was achieved in traditional approaches that made use of database integrity constraints to implement the various business rules.

5 A Form-based Approach

The previous sections discussed integrity aspects concerning relational database and XML approaches. From this it is clear that the adherence to semantic integrity is of principal importance. The consideration of semantic integrity aspects will form an integral part in achieving the main objective of this article. Another aspect that also requires consideration is usability.

A number of known good practise usability guidelines exists. However, seeing that usability goes hand in hand with the study of human behavior, this is not included in the proposed solution to the problem at hand. Instead, a rule-based approach is suggested where the usability aspects depend on users inputs.

For XML to reform from a text-based approach to a more user-friendly approach, a model must be devised to produce usable form-based input forms. Aspects this model must be able to implement is semantic integrity that is the result of technical aspects, but also the result of adhering to usability rules declared by the user. Another feature that this proposed model requires is that it must be able to make use of information obtained/known of an XML document, to produce these usable interfaces. For this it can be seen that the use of XML schemas, that contain enough information to adhere to all of these model requirements, plays a critical part in the design of such a model. Figure 3 depicts the working of such a model.

It can be seen that the model takes a schema and user declared rules as input. Some form of transformation is done on the inputs. This in turns generates the required output. The output can be an HTML form, C# code,

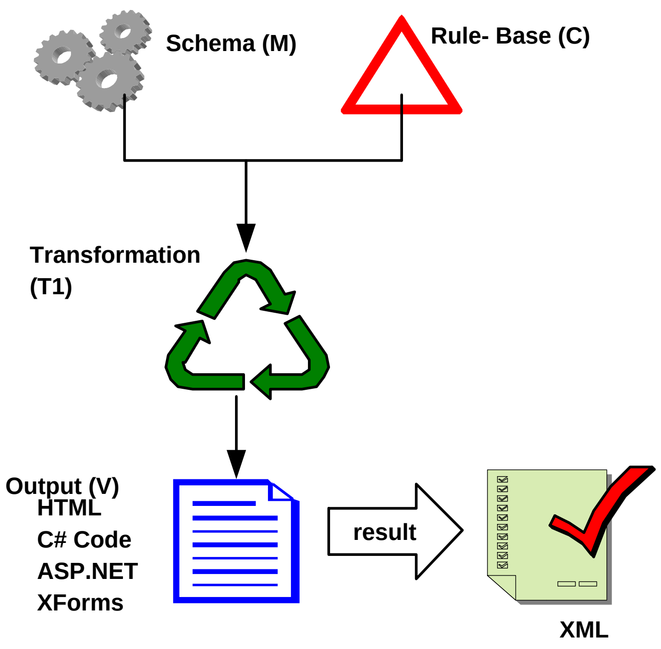

Figure 3: Overview of Proposed Model

ASP.NET code, or even syntax for XForms. The interaction of users with these outputs results in valid XML documents that conform to a high level of integrity. Integrity would be achieved, since XML schema declarations are used to generate the input forms (source code).

This approach is similar approach as what is used in the Model View Controller (MVC) architecture in the Java 2 Standard Edition. This architecture separates the presentation of data (`Output`) from the underlying data representation (`Schema`) and the control logic (`Rule-Based`) for that data (Deitel, Deitel, & Santry, 2002).

This section provided a conceptional explanation of a proposed model that can be used to automatically convert (depending on inputs) a text-based XML approach to a form-based approach. An overview will now be be provided of current work towards the successful implementation of this model.

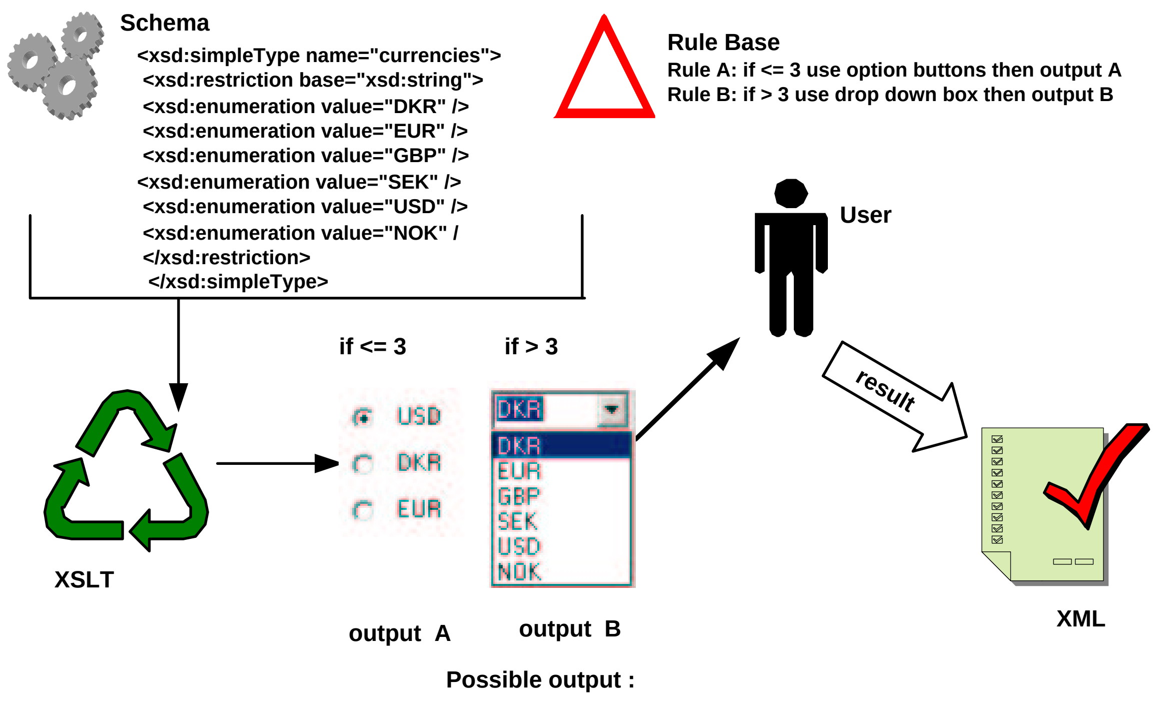
Schema
<xsd:simpleType name="currencies">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="DKR" />
<xsd:enumeration value="EUR" />
<xsd:enumeration value="GBP" />
<xsd:enumeration value="SEK" />
<xsd:enumeration value="USD" />
<xsd:enumeration value="NOK" /
</xsd:restriction>
</xsd:simpleType>
Rule Base
Rule A: if <= 3 use option buttons then output A
Rule B: if > 3 use drop down box then output B
User
if <= 3
if > 3
USD
DKR
EUR
DKR
DKR
EUR
GBP
SEK
USD
NOK
result
XSLT
output A
output B
XML
Possible output :

Figure 4: Overview of Current Work

6 Current Work

Currently, work is focussed on producing HTML input forms using a similar approach as explained in the previous section. At the moment the generation of HTML forms, using an XML schema, user declared rules as inputs and using an XSLT processor to perform transformation, is being investigated.

Explanation of this concept is done at the hand of an example. The example makes use of the schema discussed in section 4.2.2. Figure 4 provides a graphical depiction of this current work.

It can be seen in figure 4 that a schema and user defined rules are taken for inputs. The transformation considers the rules and depending on the underlying schema produces stylesheets. These stylesheets are assigned to underlying data extracted from the schema declaration. An XSLT processor transforms this assignment to the appropriate HTML input form.

An example of the proposed output of such transformations can be seen by looking at the various rules that are declared and the subsequent outputs. If rule A was satisfied, which means that only up to three enumeration elements exists, the subsequent output is output A. Otherwise if more than three enumeration entries existed the output generated is output B. This also has a added benefit that code generated caters for various output devices. For example if the output device used is a PDA a dropdown box can be used instead of option buttons to cater for a large number of options.

7 Related Work

XForms is the W3C's name for a candidate recommendation (latest version 12 November 2002) of web forms that can be used with a variety of platforms. XForms comprises of parts which respectively describe what the form does and how it looks. It also makes use of the Java Model View Controller architecture to keep the data logic separate from the control and presentation aspects.

A similar example as in section 6 can be given by using XForms form controls, namely select and select1. These controls provides an abstraction that the form must adhere to. The use of the select control specifies that the user is only allowed to select none, one or many options. The select1 form control again allows the selection of only one option from the user. These forms controls can then be associated with drop down boxes, radio buttons, or any other traditional form input methods (Dubinko, Merrick, Raman, & Klotz, 2002).

XForms can thus be considered as an alternate output for the transformation process. Similarly the "Views" project (Bishop & Horspool, 2002) expresses GUI forms in terms of XML. A correspondence between the proposed model and the views architecture may require some further investigation.

8 Conclusion

The inherent features provided by XML made it a popular choice for computer and human users. However, human users are not satisfied with working with large amounts of text, instead these users require user interfaces to interact with the systems they are working on. This is even more true in an environment where users are required to perform input on XML text.

This article introduced a proposed model that is capable of transforming XML to form-based input forms, with focussed placed on integrity. These input forms adhere to usability aspects as defined by users. With architectures like for example the Sun's Java Model View controller, it is now possible to produce implementations that have central underlying data, but different views and processing logic on this data. The fact that the underlying data is never changed allows the "new" form-based input forms to retain the various XML inherent features.

References

Bishop, J., & Horspool, N. (2002). Views. (`http://www.cs.up.ac.za/~jbishop/rotor/`)

Bray, T., Paoli, J., Sperberg-Mcqueen, C., & Maler, E. (Eds.). (2000). *Extensible Markup Language: XML 1.0 (Second Edition).* `http://www.w3.org/TR/2000/REC-xml-20001006`: W3C Recommendation.

Damiani, E., Vimercati, S. de Capitani di, Paraboschi, S., & Samarati, P. (2000). Securing XML documents. In *Proceedings of the 6th international conference on extending database technology: Advances in database technology - EDBT 2000.* Konstanz, Germany: Springer.

Deitel, H. M., Deitel, P., & Santry, S. (2002). *Advanced Java 2 Platform.* New Jersey: Prentice Hall.

Dick, K. (2000). *XML a managers guide.* International: Addison Wesley.

Donatello, J. (Ed.). (1989). *Information Processing Systems - Open Systems Interconnection Basic Reference Model - Part 2: Security Architecture.* `http://www.ISO.ch/ISO7498-2`: ISO Standard.

Dubinko, M., Merrick, R., Raman, T. V., & Klotz, L. L., Jr (Eds.). (2002). *XForms 1.0.* `http://www.w3.org/TR/2002/CR-xforms-20021112/`: W3C Candidate Recommendation.

Gropp, E. (2003, February). *Transforming XML Schemas.* `http://www.xml.com/pub/a/2003/01/15/transforming-schemas.html`.

Leyman, F., & Roller, D. (2000). *Production Workflow concepts and techniques.* New Jersey: Prentice-Hall.

Manola, F. (1999). Technologies for a Web Object Model. *IEEE Internet Computing*, 38–46.

Motro, A. (1998). Integrity = Validity + Completeness. *ACM Transactions on Database Systems*, *14*, 480–502.

North, K. (2000). *Database Magic with Ken North.* New York: Prentice Hall.

A technical introduction to XML. (1998, October). `http://www.XML.com/pub/a/98/10/guide0.html`.

The financial assistance of National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the National Research Foundation.