**XML Document** **Transformation for** **Data Manipulation Operations**

 Aigul Mukhitova
*al-Farabi Kazakh National*
*University*Almaty, Kazakhstan
mukhitova.aigul@gmail.com

Aigerim Yerimbetova
*Institute of Information and Computational Technologies*Almaty, Kazakhstan
aigerian@mail.ru

Nenad Mladenovic
*Matematical Institute, Serbian Academy of Sciences and Arts*
Belgrade, Serbia
nenadmladenovic12@gmail.com

*Abstract*—Integration of information resources to provide users with access to miscellaneous information and the ability of information management is carried out through specialized adaptive administration and user graphic web interfaces. In this paper we study the technology of creating an adaptive graphical editor of XML records to provide access to distributed heterogeneous information resources through graphical user WEB-interfaces. The methodology for creating user interfaces suitable for the structure and functionality of information sources is also described herein. The proposed XML record editor is able to visualize the full structure of an XML document as a screen form of an HTML page. The program is a server-side WEB application that provides on-screen forms for creating and editing XML documents in accordance with the selected XSD schema.

Keywords—adaptive web interfaces, data manipulation operations, XML document transformation, XSLT transformation.

# Introduction

The main goal of integration of information resources is to combine them at the physical or virtual level into a common information space to provide users with access to heterogeneous information and with the ability of information management. This is done with the help of specialized adaptive administrative and user graphical web interfaces capable of adjusting to the structure and functionality of information resources.

The development of adaptive user interfaces is the development of client-server applications that allow easy data exchange between programs and can be located anywhere, be written in any language and run under any operating system and on any hardware platform [1].

The best way to represent structured information is to visualize it as an on-screen form of an HTML page. XML and HTML are a set of SGML, so they have basic principles [2]. The structure of an XML document is similar to a regular HTML page. XML is an easy-to-read and easy-to-document format with no limit on document length. XML is more dynamic and allows for easy generation of new data schemas and transition rules between them, which are formulated in the same language (XSLT-transformation), and also supported by a large number of software manufacturers. Information from an XML document can be represented as an HTML sheet by simple manipulation. HTML forms and XML elements are logically equivalent to content information. Thus it automatically converts any software for any XML data into a web browser. The embedded HTML form which acts as a data entry form is in fact a graphical user interface because it is a simple XML editor for web programming. During processing XML files, a problem of availability of most programs has occurred. When you choose tools for editing XML data as well as editing an XML document, you should take the problem into the attentive consideration.

Thus, the relevant task is to research and apply adaptive technologies to develop user and administrative interfaces that provide the ability to manage heterogeneous data (data creation and modification).

The purpose of the work is to create a set of engineering solutions and techniques which implement a systematic and comprehensive approach to the technology of building adaptive graphical web interfaces. The method of research is based on a systematic analysis of modern technologies to create adaptive graphical web interfaces, development of the own technology and its experimental testing.

# Related Works

A number of research works have been carried out in the field of modeling, storing, manipulating XML data, their indexing and methods of queries. Conventionally, these studies can be divided into main groups according to their functional purpose [3]:

* XML data queries [4-10];
* XML conceptual modeling [11,12];
* XML data clustering [13];
* XML streaming [14];
* data storage and query processing in XML

databases [15];

* manipulation in conventional and temporal XML databases [16-19];
* transformation of XML documents into the OWL ontology [20, 21].

By simple manipulations, information from an XML document can be presented as an HTML page. HTML forms and XML elements are logically equivalent in terms of information content that makes it possible to automatically convert them via a web browser using a single software tool for any XML data [22]. The generated HTML form, which acts as a data entry form, is essentially an adaptive graphical interface, as it is a simple low-level XML editor based on web programming. With the advent of XML, which brought many convenient features for software system developers, the problem of editing XML files also appeared [23]. There was a need to study the problem that occurs while editing XML data and choosing tools that allow you to make changes to the XML document.

Thus, the task of research and application of adaptive technologies for the development of user and administrative interfaces that provide the ability to manage heterogeneous data (data creation and modification) is urgent.

Russian and foreign scientists have made a great contribution to the research of designing, implementing, and maintaining administrative and user interfaces that support various types of dialog [24-29]. In view with the above, research aimed at solving the problems of reducing the complexity of designing, prototyping, implementing and maintaining the user interface is relevant. Based on modern achievements in this field, the main paradigms for automating user interface development have been formed.

The purpose of the research was to study and develop methods and software tools for building adaptive administrative and user interfaces that increase the degree of functionality and expand the possibilities of user interaction during the search and edition of heterogeneous information in a distributed information system.

**The scientific novelty** of the work is to create a set of technical developments and techniques (models and algorithms) that implement a systematic and integrated approach to building adaptive user interfaces.

**The theoretical value** of the work is to provide a formal description of the developed adaptive model for entering and editing metadata presented in XML format, based on the use of XSD data schema definitions using XSLT technologies and its place in the architecture of a web server application.

**The practical value** of this work is to develop a server-side Web application that provides on-screen forms for creating and editing XML documents in accordance with the selected XSD schema. Screen forms are generated on the server side and provided to the user as HTML pages. Completed on-screen forms are processed by the server to generate an XML document. This software product can be used as an independent

manipulation of XML data

# The Methodolofy of Creating Adaptive Graphic Web-Interface

HTML forms and XML elements are logically equivalent in terms of information content. Therefore it is possible to convert them automatically through a web browser using a unified software tool for any XML data. The generated HTML form, which acts as a data entry form, is essentially a simple low-level web-based XML editor, and is an adaptive graphical interface [13].

The process of converting XML documents from one XML schema to another can be separated in three main parts [2, 30, 34-35]:

* Creating XML code - creating code using ordinary text or specialized XML code editors. Many DBMS are equipped with a function of direct generation of XML code.
* Generation of style XSL-tables for processing and displaying an XML-document. XSL tables are to be generated by an administrator-programmer.
* Imposition of XSL style sheets. Style XSL-tables are imposed on XML-document according to certain formatting rules and the result is a document corresponding to the client's request in the following output formats: HTML, PDF, WML and XML.

*Client-side actions*

Conversion of XML documents by imposition of style XSL tables is usually to be done on the client side. The stylesheet is loaded only once, reducing the load on the server. Based on the principle of overlaying a CSS file with an HTML file, browsers can attach styles to XML documents and generate the final document.

*Server-side actions*

If browsers do not understand full-fledged HTML, and cannot perform xml\xsl transformations themselves, then the style application to an XML document can be performed by the server in two ways [14]:

* manually - compile xml-files/stylesheets and locate the resulting files on the server;
* automatically - the software framework for Web application development (CMF) performs the required xml-document transformation and transmits it to the user: at the request of Acrobat Reader - the published page will come in PDF format, on request of a WAP device - the client will receive WML page, Web-browsers will receive an HTML/XHTML page.

In our study, we propose the following approach for representing structured information in XML format through ADAPTIVE GRAPHIC WEB-INTERFACE. The rules for constructing the structure of an XML record in terms of XSD [2] is an XML structure that can be processed using standard XSLT tools.

In the process of transforming an XML document for manipulation operations, it is necessary when retrieving a record from a specific source to have a complete description of the possible structure of the retrieved record.

The following options are possible:

* The easiest way. Get an XSD when the root XML entry element contains a URL reference to the XSD data schema in use as the value of the schemaLocation attribute when defining the namespace to use.
* Variant when the record retrieved for XML editing contains a namespace identifier (URI), but does not contain a URL reference to the XSD data schema being used. In this case, a request to the information system is required to provide the XSD by the namespace identifier.
* The option is more difficult when the entry received for XML editing does not contain namespace definitions. In this case, you need to make a request to the information system to provide a default XSD by the name of the information resource (database) or use an XSD that matches the schema requested when creating a request to receive data.

In any of the above cases, data modifications are required to initialize the graphical interfaces:

* + Description of the data schema in the form of an XML structure in accordance with the XSD rules.
	+ XML ​​structure containing the extracted data for editing (not required to create a new record).
	+ Description of the styles of forming the elements of the graphical interface (optional).
	+ Description of the rules for generating elements of the graphical interface in accordance with the rules of the used XSD and the values ​​of the elements of the edited XML record. However, these rules can be XSLT-transformation rules applied to XSD.

As an illustration, Fig. 1 shows the algorithm of the adaptive XML records editor in the client-server architecture, embedded in the WEB server of the ZooSPACE platform (ZooSPACE-W). The XML editor itself corresponds to the area enclosed by the dashed line for the server side. For the client part, a ready-made HTML form is provided for entering and (or) editing data. Moreover, this form already contains all the necessary tools (java scripts) for correct data entry, including:



Fig. 1.The functional diagram of XML editor for the heterogeneous system ZooSPACE-W

# Implementation and Validation

## XML-document Transformation to HTML-form

The XSLT template describes the rules for how XML elements are converted to HTML forms. It specifies the order, nesting of elements, display rules, tooltips, or additional information which could be found both in the XSD file and in the XML document itself.

Herebelow (Fig. 2) is an example where an XSLT template is applied to an XML document.



Fig. 2.The code snippet of XSLT template

HTML code (Fig. 3) as a result of transformation which is displayed in the main editor window:



Fig. 3. HTML-code snippet after transformation

The XSLT template extracts basic attributes of elements from the XML document:

Creates an xsl: template for each element with the match

attribute equivalent to the element name:

<xsl:template match="//xs:element">...

 </xsl:template>

<xsl:template match="//xs:complexType">...

 </xsl:template>

<xsl:template match="//xs:simpleContent">...

 </xsl:template>

<xsl:template match="//xs:annotation">...

 </xsl:template>

<xsl:template match="//xs:appinfo">...

 </xsl:template>

<xsl:template match="//xs:documentation">...

 </xsl:template>

<xsl:template match="//xs:extension">... </xsl:template>

If a complex element is processed, the xsl:apply-templates select="\*" function with the appropriate parameters is to be added to the template.

The correct XSLT functions and elements are to be added to the xsl:template so that xsl:template could generate the necessary input elements if required.

A new XML-document is created by pressing the "Record" button. The HTML form is checked against the pre-built data model, and if no errors are found a new XML-file is generated (Fig. 4).

The main task is as follows:

1. Create an xsl:template for each element with a match attribute equivalent to the element name;

2. If it is a datatype element, it is required to add it as well as functions necessary for creation of the element's value to xsl:template;

3. If it is a complex element, it is required to add the function xsl:apply-templates select="\*";

4. Create xsl:template for each element with match attribute which is equal to insert-elementName, where elementName – the Element Name;

5. If it is a complexType element, it is necessary to add recursively all its descendants N times, where N is the value of the minOccurs attribute for the descendant element.

The method for how to add an element that is missing in the minimal correct document is described below. There are different options for action [15, 16]:

1. When creating a new document, add all elements to that document, including elements for which minOccurs is equal to zero. This is a fairly simple solution, and although the resulting document is correct, it doesn't quite meet our expectations: an empty element is not equivalent to the absence of an element in the document.

2. To correct XSLgui so that it could generate so generates a " + " sign after each element. A document with a large number of elements will be inconvenient for the user, because it will take too many attempts to find the correct place to insert the element.

3. Find a way to check the conciseness and correctness of the pluses, and pass this information to XSLgui. Conceptually this is an elegant solution, but technically it is quite difficult to implement.

4. Another solution is to assign pluses to the descendants of a complexType element as described below. For all descendants with minOccurs equal to 0, and if the number of occurrences of the element in the document is zero, create a "+" sign at the level of the complexType-element containing the descendants.

An XSLT template identifies the main elements from an XML document:

* plain text;
* comments;
* element attributes;
* a complex element, which may have a value, attributes and necessarily contains other elements inside;
* a simple element, has only a value and attributes.

When generating data-entry fields, the type of data and restrictions imposed should be taken into account. In particular, the input field for elements and attributes will be a drop-down list of values (fig.10) if there are XSD type definitions:

When generating data entry fields, the data type and the restrictions imposed are taken into account. In particular, the input field for elements and attributes will be in the form of a drop-down list of values (Figure 4) if there are XSD type definitions:

<xsd:simpleType name="recordTypeType">

 <xsd:restriction base="xsd:NMTOKEN">

 <xsd:enumeration value="Bibliographic"/>

 <xsd:enumeration value="Authority"/>

 <xsd:enumeration value="Holdings"/>

 <xsd:enumeration value="Classification"/>

 <xsd:enumeration value="Community"/>

 </xsd:restriction>

</xsd:simpleType>



Fig. 4.XML record graphical editor: data entry fields

If an XSD element contains a template reference (RegEx), for example:

<xsd:simpleType name="indicatorDataType" id="ind.st">

 <xsd:restriction base="xsd:string">

 <xsd:pattern value="[\da-z ]{1}"/>

 </xsd:restriction>

</xsd:simpleType>

then the edit form generates a call to the function to verify that the template matches the input data, i.e. the XSLT code will be executed:

. . .

<xsl:for-each select="xsd:simpleType/xsd:restriction/xsd:pattern">

 <xsl:attribute name="onChange">

 <xsl:text>e\_change(this, /</xsl:text>

 <xsl:value-of select="@value"/>

 <xsl:text>/);</xsl:text>

 </xsl:attribute>

</xsl:for-each>

 . . .

Which will generate elements of form

<input type="text" onChange="e\_change(this, /[\da-z ]{1}/);" . . . />

The described approach to XSD-based XML generation introduces the problem of recursive definitions. The recursiveness can occur when using references to types and names. Also, the described approach to the formation of XML based on XSD raises the problem of recursive definitions. Recursiveness can occur when using references to types and names [1,2]. For the following snippet of schema definition using XSD

<xsd:complexType name="organization">

 <xsd:sequence>

 <xsd:element name="id" type="int"/>

 <xsd:element name="name" type="string"/>

 <xsd:element name="sub-org" type="tns:organization"/>

 </xsd:sequence>

</xsd:complexType>

<xsd:element name="region">

 <xsd:complexType>

 <xsd:sequence>

 <xsd:element name="id" type="int" />

 <xsd:element ref="tns:region" />

 </xsd:sequence>

 </xsd:complexType>

</xsd:element>

<xsd:element name="record">

 <xsd:complexType>

 <xsd:sequence>

 <xsd:element name="id" type="int" />

 <xsd:element name="organization" type="tns:organization" />

 <xsd:element ref="tns:region" />

 </xsd:sequence>

 </xsd:complexType>

</xsd:element>

XML elements with unlimited XPath length are possible

/record/organization/sub-org/sub-org/sub-org . . .

/record/region/region/region/region . . .



Fig. 5. Recursion fragment

To avoid infinite nesting of elements when generating graphical editing interfaces for records, you can use the control of the number of nestings and limit them according to the current need.

XSD processing with XSLT takes into account various features of XSD definitions, including references and recursion. The data entered is automatically saved in the same database from which the record was retrieved.

# Conclusion

The proposed technology of building adaptive graphical WEB-interfaces allows for effectively solving the problem of integrating heterogeneous information resources based on the database schemes of XSD application using XSLT transformations. The described methodology has quite general ways of use and can be applied to build adaptive graphical WEB-interfaces that allow generating HTML-forms for data input and editing.

The implementation of the above concepts provides for guaranteed interaction of client and server applications on different platforms when working in heterogeneous environments.

The proposed adaptive graphical XML record editor allows for importing any XML data and transforming its structure in efficient and simple way, while transformation of source data of any structure without any changes in the program code is possible within the same processing.

Currently, the adaptive graphical XML record editor is implemented in the ZooSPACE-W subsystem of the ZooSPACE platform of the Institute of Computational Technologies, Siberian Branch of the Russian Academy of Sciences (Novosibirsk). Work is underway to improve the editor's functionality in terms of extending the list of supported XSD and JSON elements.

The field of application of the above technology is the integration of the described adaptive graphical web-interface in a distributed information system in order to increase the efficiency of user access to heterogeneous information and the ability to manage it.

The program can be used as an instantiated programming unit to generate and edit data represented in XML format, as well as a plug-in module in different server software for heterogeneous information integration systems.

##### Acknowledgment

This research has is funded by the Science Committee of the Ministry of Education and Science of the Republic of Kazakhstan (Grant No. AP08857179).

##### References

1. O. Zhizhimov, “Explain services on ZooSPACE platform and adaptive user interfaces**”,** CEUR Workshop Proceedings, vol.1536, pp.30 – 36, 2015.
2. W3C, Extensible markup language (XML) 1.0 (5th edition), W3C recommendation, 2008, <https://www.w3.org/TR/2012/>WD-xml-proc-profiles-20120124/#background//. (Accessed 20 April 2021).
3. Z. Brahmia, H. Hamrouni, R. Bouaziz, “XML data manipulation in conventional and temporal XML databases: A survey”, Computer Science Review Volume 36, May 2020, 100231, https://doi.org/10.1016/j.cosrev.2020.100231
4. T.S. Chung, H.J. Kim, “Techniques for the evaluation of XML queries: a survey”, Data Knowl. Eng., Volume 46 (2), 2003, pp. 225-246.
5. G. Gou, R. Chirkova, “Efficiently querying large XML data repositories: A survey”, IEEE Trans. Knowl. Data Eng., Volume 19 (10), 2007, pp. 1381-1403.
6. N. Bidoit, D. Colazzo, N. Malla, C. Sartiani, “Evaluating queries and updates on big XML documents”, Inf. Syst. Front, Volume 20 (1), 2018, pp. 63–90.
7. L. Anselma, L. Piovesan, A. Sattar, B. Stantic, P. Terenziani, “Representing and querying now-relative relational medical data”, Artif. Intell. Med. Volume 86, 2018, pp. 33–52.
8. F. Wang, C. Zaniolo, “Temporal queries and version management in XML-based document archives”, Data Knowl. Eng, Volume 65 (2), 2008, pp. 304–324.
9. F. Wang, C. Zaniolo, X. Zhou, “ArchIS: an XML-based approach to transaction-time temporal database systems”, VLDB J. Volume 17 (6), 2008, 1445–1463.
10. N. Bidoit, D. Colazzo, C. Sartiani, A. Solimando, F. Ulliana, “Andromeda: A system for processing queries and updates on big XML documents”, T. Morzy, P. Valduriez, L. Bellatreche (Eds.), New Trends in Databases and Information Systems: ADBIS 2015.
11. M. Necasky, “Conceptual Modeling for XML: A Survey”, Proceedings of the DATESO 2006 Annual International Workshop on DAtabases, TExts, Specifications and Objects, Desna, Czech Republic, 26–28 April, 2006.
12. H. Chen, H. Liao, “A survey to conceptual modeling for XML”, Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT’10, Chengdu, China, 9–11 July, 2010, pp. 473–477.
13. A. Algergawy, M. Mesiti, R. Nayak, G. Saake, “XML Data clustering: An overview”, ACM Comput. Surv., Volume 43 (4), 2011, p. 41.
14. X. Wu, D. Theodoratos, “A survey on XML streaming evaluation techniques’, VLDB J., Volume 22 (2), 2013, pp. 177-202.
15. S.C. Haw, C.S. Lee, “Data storage practices and query processing in XML databases: A survey”, Knowl.-Based Syst., Volume 24 (8), 2011, pp. 1317-1340.
16. S. Faisal, M. Sarwar , “Temporal and multi-versioned XML documents: A survey”, Inf. Process. Manag., Volume 50 (1), 2014, pp. 113-131.
17. Z. Brahmia, R. Bouaziz, “Data manipulation in multi-temporal XML databases supporting schema versioning”, Proceedings of the 4th In-ternational EDBT Workshop on Database Technologies for Handling XML Information on the Web, DaTaX’09, Saint-Petersburg, Russia, 22 March, 2009, p. 14.
18. H. Hamrouni, F. Grandi, Z. Brahmia, “Deferred repair of inconsistencies resulting from retroactive updates of temporal XML monetary data”, Int.J. Web Inf. Syst. Volume 13 (4), 2017, pp. 485–519.
19. J. Clifford, A. Croker, F. Grandi, A. Tuzhilin, “On Temporal grouping”, Proceedings of the International Workshop on Temporal Databases, Zürich, Switzerland, 17–18 September, 1995, pp. 194–213.
20. J. Clifford, C.E. Dyreson, T. Isakowitz, C.S. Jensen, R.T. Snodgrass, “On the semantics of now in databases”, ACM Trans. Database Syst. Volume 22 (2), 171–214.
21. M. Hacherouf, S.N. Bahloul, C. Cruz, “Transforming XML documents to OWL ontologies: A survey”, J. Inf. Sci., Volume 41 (2), 2015, pp. 242-259.
22. XSL Transformations (XSLT) Version 2.0: W3C Recommendation, http://www.w3.org/ TR/xslt20. (Accessed 15 March 2021).
23. K. Luyten, M. Abrams, J. Vanderdonckt, Q. Limbourg, “Developing User Interfaces with XML: Advances on User Interface Description Languages,” Similar Noe : Gallipoli, Proceedings of Satellite Workshop of Acm International Conference on Advanced Visual Interfaces, 167 pages, May 2004.
24. A. Grinberg, “Modifying XML, in: XML and JSON Recipes for SQL Server”, Apress, Berkeley, CA, USA, 2018, pp. 135–156.
25. J. Liu, X.X. Zhang, “Dynamic labeling scheme for XML updates”, Knowl.-Based Syst, Volume 106, 2016, pp. 135–149.
26. Z. Brahmia, F. Grandi, R. Bouaziz, “τXuf: A temporal extension of the XQuery update facility language for the τXSchema framework”, Proceedings of the 23rd International Symposium on Temporal Repre-sentation and Reasoning, TIME’16, 17–19 October, Technical University of Denmark, Copenhagen, Denmark, 2016, pp. 140–148.
27. H. Hamrouni, Z. Brahmia, R. Bouaziz, “A systematic approach to efficiently managing the effects of retroactive updates of time-varying data in multiversion XML databases”, Int. J. Intell. Inf. Database Syst, Volume 11 (1), 2018, pp. 1–26.
28. Z. Brahmia, H. Hamrouni, R. Bouaziz, “Tempox: A disciplined approach for data management in multi-temporal and multi-schema-version XML databases”, J. King Saud Univ. Comput. Inf. Sci., 2019.
29. M.-A. Baazizi, N. Bidoit, D. Colazzo, N. Malla, M. Sahakyan, “Projection for XML update optimization”, Proeedings of the 14th International Con-ference on Extending Database Technology, EDBT’11, Uppsala, Sweden, 21–24 March, 2011, pp. 307–318.
30. B. Bouchou, D. Duarte, M.H.F. Alves, D. Laurent, M.A. Musicante, “Schema evolution for XML: A consistency-preserving approach”, Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science, MFCS’04, Prague, Czech Republic, 22–27 August, 2004, pp. 876–888.
31. A. Davoudian, L. Chen, M. Liu, “A survey on NoSQL stores”, ACM Comput. Surv. Volume 51 (2), 2018, Article 40.
32. J. Pokorný, “NoSQL databases: a step to database scalability in web environment”, Int. J. Web Inf. Syst. Volume 9 (1), 2013, pp. 69–82.
33. S. Tiwari, “Professional NoSQL”, John Wiley & Sons, Inc., Indianapolis, Indiana, USA, 2011.
34. Z. Brahmia, F. Grandi, B. Oliboni, R. Bouaziz, “Schema change operations for full support of schema versioning in the τXschema framework”, Int. J. Inf. Technol. Web Eng. Volume 9 (2), 2014, pp. 20–46.
35. F. Currim, S. Currim, C.E. Dyreson, R.T. Snodgrass, “A tale of two schemas: creating a temporal XML schema from a snapshot schema with τXSchema”, Proceedings of the 9th International Conference on Extending Database Technology, EDBT’04, Heraklion, Crete, Greece, 14–18 March, 2004, pp. 348–365.
36. C.E. Dyreson, F. Grandi, “Temporal XML”, L. Liu, M.T. Özsu (Eds.), Encyclopedia of Database Systems, second ed., Springer-Verlag, New York, USA, 2018, <http://dx.doi.org/10.1007/978-1-4614-8265-9_411>.
37. R.T. Snodgrass, C.E. Dyreson, F. Currim, S. Currim, S. Joshi, “Validating quicksand: Schema versioning in τXschema”, Data Knowl. Eng.Volume 65 (2), 2008, pp. 223–242.
38. ANSI/NISO Z39.50-2003. Information Retrieval (Z39.50): Application Service Definition and Protocol Specification. - NISO Press, Bethesda, Maryland, U.S.A. — Nov 2002. — ISSN: 1041-5653. — ISBN: 1-880124-55-6.