The ability of the end user to work with a large amount of data from a large number of heterogeneous sources and at the same time get an effective result from the work is carried out through the use of graphical web interfaces built on the basis of XML technologies that allow displaying any structure of a file presented in XML format. As a data exchange method between applications on the Web, XML still lacks capabilities for identification of web resources and a system that uses them, and capabilities to express the knowledge provided by XLM documents. In this study, a web interface has been developed (a web-based server application), as an XML records editor that provides display forms for the creation and editing of XML documents and is able to adapt to the internal resources of the system used. The technology is based on the XSD data set schema transformation by the way of XSLT transformations. Screen forms are generated on the server side and are provided to the user with all the necessary tools for correct input and/or editing of heterogeneous data. A distinguishing characteristic of this technology is the ability to display both properly and improperly formed XML data. The developed graphical interface allows any application to automatically exchange and read information from other applications without human intervention, which significantly improves performance and ease of use. This software solution could be used both as an independent data building and editing module presented in the XML format, and as a built-in module plugged into various server software for heterogeneous information management systems

Keywords: web interfaces, XML document management/navigation, XML/XSLT transformation, non-well-formed data

UDC 004.62;65
DOI: 10.15587/1729-4061.2023.276585

DEVELOPMENT OF AN ADAPTIVE GRAPHIC WEB INTERFACE MODEL FOR EDITING XML DATA

Aigul Mukhitova
Doctoral Student*
Senior Lecture
Department of Information Technologies**
Aigerim Yerimbetova
Corresponding author
PhD, Associate Professor, Leading Researcher*
Professor
Department of Software Engineering
Institute of Automation and Information Technologies
Satbayev University
Satbayev str., 22, Almaty, Republic of Kazakhstan, 050013
E-mail: aigerian@mail.ru
Lyailya Cherikbayeva
PhD, Acting Associate Professor
Department of Computer Science**
*Institute of Information and Computational Technologies
Shevchenko str., 28, Almaty,
Republic of Kazakhstan, 050010
**Al-Farabi Kazakh National University
Al-Farabi ave., 71, Almaty, Republic of Kazakhstan, 05059

Received date 03.02.2023
Accepted date 11.04.2023
Published date 28.04.2023

***How to Cite**: Mukhitova, A., Yerimbetova, A., Cherikbayeva, L. (2023). Development of an adaptive graphic web interface model for editing XML data. Eastern-European Journal of Enterprise Technologies, 2 (2 (122)), 26–35. doi: https://doi.org/10.15587/1729-4061.2023.276585*

1. Introduction

The main task of integrating information resources is to combine them at the physical or virtual level into a single information space to provide users with access to heterogeneous information and the ability to manipulate it. This is done using specialized adaptive administrative and user graphical web interfaces that can adapt to the structure and functionality of information resources.

Such interfaces can be developed using various data presentation formats EDI, CSV, XML, JSON), domain-specific language and common languages (XSLT, SQL, Java, C#). This diversity helps to choose the most optimal data format or language based on specific requirements. A possible solution for this problem can be found in the development of platform-independent specifications that will be used for the source generation for each required platform.

Over the past decades, a number of studies on this topic have been carried out, which can be conditionally classified into three individual groups:

– data storage and request processing in statistical and temporary XML databases;

– XML document navigation and transformation;

– conceptual modeling and management of the XML document;

– clustering and streaming XML data.

In terms of application, all the above scientific research on XML data processing can be divided into the following groups:

– database Systems – Ontology-Query Processing;

– industry-Web Services-Models;

– XML-Access Control-Labeling Scheme;

– software Engineering – Models-Software Design.

With the appearance of Web 2.0, data began to spread throughout the Internet, through online and offline applications, and across all application domains. Web data is semi-structured data downloaded through browsers and applications that exchange data over HTTP Internet protocols. Basically, all web data is XML-based to provide the following services:

– for simple display of commercial information – XHTML/HTML on commercial websites;

– for instant messaging in WhatsApp, Skype, Gtalk – XMPP;

– in e-commerce of financial transactions – CDF3;

– for processing and storing medical electronic records – HL7;

– social media – Facebook, LinkedIn, Google Plus, etc. – XHTML/HTML.

XML, which was originally defined as a metalanguage, is now actively used in the Web-space as a way to exchange data between applications; it still lacks the capabilities to define web resources and the system that uses them, or the ability to express knowledge provided by XML documents.

To present structured information in the XML-format, the main point is the presence of a description of guidelines for XML-record development guidelines, i.e. data scheme description. As a rule, when it comes to XML, these guidelines are formulated in XSD terms, and they represent an XML structure, which can be processed using standard methods, such as XSLT.

Therefore, XML technologies-based graphical interfaces allow for displaying any XML-format file structure presented. When using XML in information management and information exchange strategies, there are problems associated with storing, retrieving, re-questing, indexing, and manipulating data. As a result, new problems of information modeling arise.

So, the studies of researching and applying adaptive administrative and user interface development technologies that help manage heterogeneous data (data creation and modification) are relevant. Therefore, studies that are devoted to the development of an adaptive graphic interface model are of scientific relevance.

2. Literature review and problem statement

There are various technologies, and data management languages used to process data in XML format. The study [1] presents an overview of the current state of XML data processing in conventional and temporal XML databases, and also suggests possible fields for future research on this subject. The authors of the study [2] consider data storage and document processing in registries and temporary XML databases. Based on both analyses, there are no existing solutions to create a graphical interface for displaying non-well-formed data.

The work [3] introduces an approach based on the canonical data model (CDM) through the implementation of a collection of rules that make it possible to transform XSD into an OWL ontology, which transforms both the XML file nodes and the links between these nodes to preserve the same structure. It is shown that this method is intended for the efficient management of web documents located in OWL, but the issue of the possibility of manipulating data has not been resolved.

For XPointer-based positioning (an advanced addressing tool), there is still a problem of extracting the incorrect format of the data content in an XML document, focused on extracting incorrect data in XML documents based on XPath 3.0. As a result of a deep analysis of the extract and filter nodes, an XPointer-based location system was developed that provides advanced addressing for XML documents to find and represent both properly and improperly formed data in XML documents. The result of the work is a tree-like display of the document structure in HTML. This allows you to see the full structure of the retrieved document, but does not solve the issue of the ability to manipulate data.

The paper [4] discusses an approach to generating XML document validators based on UML models with Object Constraint Language (OCL) rules. The authors propose a generalized chart to create validators on the basis of an approach based on the model of converting OCL constraints into XPath assertions. The transformation is implemented at the model level in the Query/View/Transformation language. Some can be generated from one of the supported platform-independent models, the data model of the Eurasian Economic Union or ISO 20022, while others can be embedded in XML Schema 1.1, XSLT, or Java. This validator allows you to check the conformity of an XML document to a specific UML diagram of a given OCL annotation, but without the possibility of changing the data values in the document itself.

Temporal XML databases have been proposed to study the complete history of data changes over time. For data modification, the author of the paper [5] suggests using XML-oriented visual languages that formally define their graphic and language syntax, as well as visual systems/tools. The described methods are advisory in nature without a practical result. In [6], XML uses a temporal data grouping estimate that considers the most likely and efficient representations of temporal information. The authors [7] propose a new schema versioning control method in τXSchema, which is a platform for creating and validating temporary XML documents. This method allows for complete, integrated, and secure schema change control. It defines their operational semantics and provides a complete and reliable set of variance primitives and a set of operations to serve each component of the τXSchema schema. In subsequent studies, the same authors [8] propose a general approach called TempoX (Temporal XML) for manipulating data in multi-temporal and multi-versioned XML databases. They defined a new multi-temporal XML data model supporting temporal schema versioning called TempoXDM (Temporal XML Data Model). In the first case, the authors solve the problem of collecting and storing data that changes over time, without the possibility of correcting the data and storing it. In the second, the authors provide specifications for the basic data manipulation operations: "insert", "replace", "develop" and "delete", but only for well-formed documents.

Editing an XML document can be a simple operation for the data operator, for example, when editing directly through a text editor, but such simplicity is unsystematic, as indicated in [9]. From this, we can draw the following conclusion: the preservation and control of the correctness of information about the structure of the document when editing data, that is, automatic validation and correction of documents transparent to the user, require different approaches. For example, editing through a graph representation of a document (Arbortext Editor (https://www.ptc.com/en/products/arbortext), XmlGrid.net (https://xmlgrid.net/), CAM XML Editor (https://camprocessor.sourceforge.net/wiki/)), or editing through the presentation of the document in the form (Oxygen XML Editor (https://www.oxygenxml.com/), EditiX (https://www.editix.com/)). All of the above editors are stand-alone tools for creating and editing XML documents without the ability to adapt to the internal resources of the system.

The paper [10] considers the automatic transformation of queries in accordance with the schema update. The authors first defined the ShEx schema update operations and then proposed an algorithm for converting a given query into a new query in accordance with the schema update. It supports TempoXUF and allows end users to manipulate (i. e. insert, replace, delete, and modify) multi-temporal XML data in the context of schema versioning through a GUI. The

conversion algorithm works well, but it is intended only for well-formed documents.

The need for flexible development of software is steadily growing due to the technological complexity of the information systems being developed. Despite the fact [11] that today's organizational knowledge is accumulated and managed using an increasing number of different technologies and methodologies, solutions for the continuous reintegration of acquired knowledge into the overall design and production process are still inadequate.

All this makes it possible to assert that it is expedient to conduct a study on the development of an adaptive graphical interface model that provides the display of forms for creating and editing XML documents in accordance with the selected XSD schema, capable of determining the web resources and systems used.

3. The aim and objectives of the study

The aim of the study is to develop a model of an adaptive graphical web interface for editing XML data. This will make it possible to edit an XML document through an automatically created graphical interface on the basis of web forms, based both on the schema of the document itself and on schemas located on other components and nodes of the system, as well as on nodes remote from the system located on the web.

To achieve the aim, the following tasks have been set:

– to develop a model technology to solve the problem of finding a structure description (XSD) for non-well-formed documents and eliminating recursive definitions;

– to experimentally implement model verification in order to analyze the effectiveness of the results in the distributed information system, based on XML/SOAP/SRW technologies.

4. Materials and methods

The object of the study is an information system based on Z39.50 using the Explain special standardized system services. The ideology of the system is based on the implementation of Z39.50 functionality in web systems in the form of XML/SOAP/SRW – ZeeRex technologies.

The main hypothesis of the study is the development of web services, taking into account the search for information based on arrays of their descriptions – metadata, including an assessment of existing solutions for representing and transforming metadata (in particular, XML).

The study was performed using the theory and practice of engineering approaches used to create distributed information systems.

The system uses a special IR-Explain-1 database where the server stores information on its configuration, databases, attributes, formats, and other entities it supports. Explain allows customers to additionally adjust to its configuration, which is the key factor in building graphical interfaces with an extensive system of self-adapting names. It was found that in this type of the existing distributed information systems, there are no subsystems for adjusting to the data structure and semantics in the form of an adaptive model for entering and editing information. An important factor in distributed systems is that the structured XML format, after transmission over the network, makes it possible to completely preserve the original structure of the record, unlike other protocols (http, ftp, etc.).

The research was performed using the system approach methodology for the analysis and synthesis of architectural solutions to create such systems and the principles of integration of these systems with external sources.

The work applied modern methods of knowledge engineering, domain engineering, methods and principles of designing problem-oriented systems, modern software development methods, and the theory of relational database building.

5. Results of experimental research on the development of an adaptive graphic interface model

5. 1. Model development technology

Currently, there are three main tools for XML documents addressing – XLink, XPath and XPointer. To initialize graphical data modification interfaces, you need the following:

– description of the data schema in the form of an XML structure in accordance with the XSD rules;

– description of the rules for generating graphical interface elements in accordance with the rules of the XSD used and the value of the elements of the edited XML record. However, these rules can act as the XSLT transformation rules applied to the XSD.

It is still a problem, how to extract the incorrect format of the data content in the XML document.

The model technology developed in this study is based on the discussion of all possible ways to obtain a complete description of the possible structure of the extracted record (XSD) (Fig. 1).

The XSD obtaining algorithm is as follows:

– Retrieve a link to the original XSD data schema as an URL as the value of the schemaLocation attribute when determining the namespace used:

```
…
$imp_file = $inp->getAttribute('schemaLocation');
        $imp_doc = new DOMDocument ( "1.0", "UTF-8" );
        $imp_doc->load ($imp_file);
…
```

If there is no link to the used XSD data schema in the form of a URL address, make an inquiry to the information system about the provision of XSD by the namespace identifier, if the XML record retrieved for editing contains the namespace identifier (URI). In our system, such an inquiry is processed by the Explain service:

```
…
$oprefix=$imp_doc->documentElement->lookupPrefix(
"http://www.w3.org/2001/XMLSchema" );
        if($oprefix != "xsd")
{
                $s1 = $imp_doc->saveXML();
                $s2 = str_replace($oprefix.":","xsd:", $s1);
                $s2 = str_replace("xmlns:".$oprefix,"xmlns:xsd", $s2);
                $imp_doc->loadXML($s2);
        }
…
```

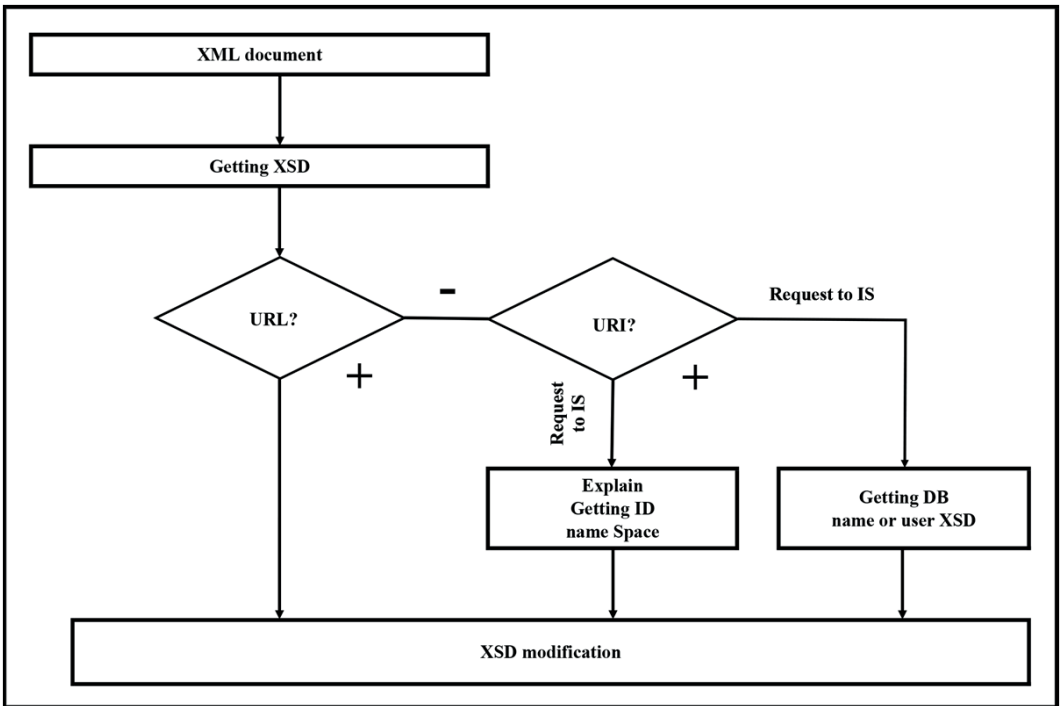

Fig. 1. Functional diagram of the model for extracting the record structure

In the cases where the XML record retrieved for editing does not have namespace definitions, or where it is impossible to use the above two methods, an inquiry must be sent to the information system for XSD by the name of information resource (database), or use the XSD that corresponded to the schema requested in the formation of the data retrieval request.

Thus, in the paper we propose an original approach, which consists in using an adaptive editor of XML records in the client-server architecture, which is built into the web server of a distributed information system. Actually, the XML editor corresponds to the area highlighted in gray, limited by the dashed line for the server part (Fig. 2).

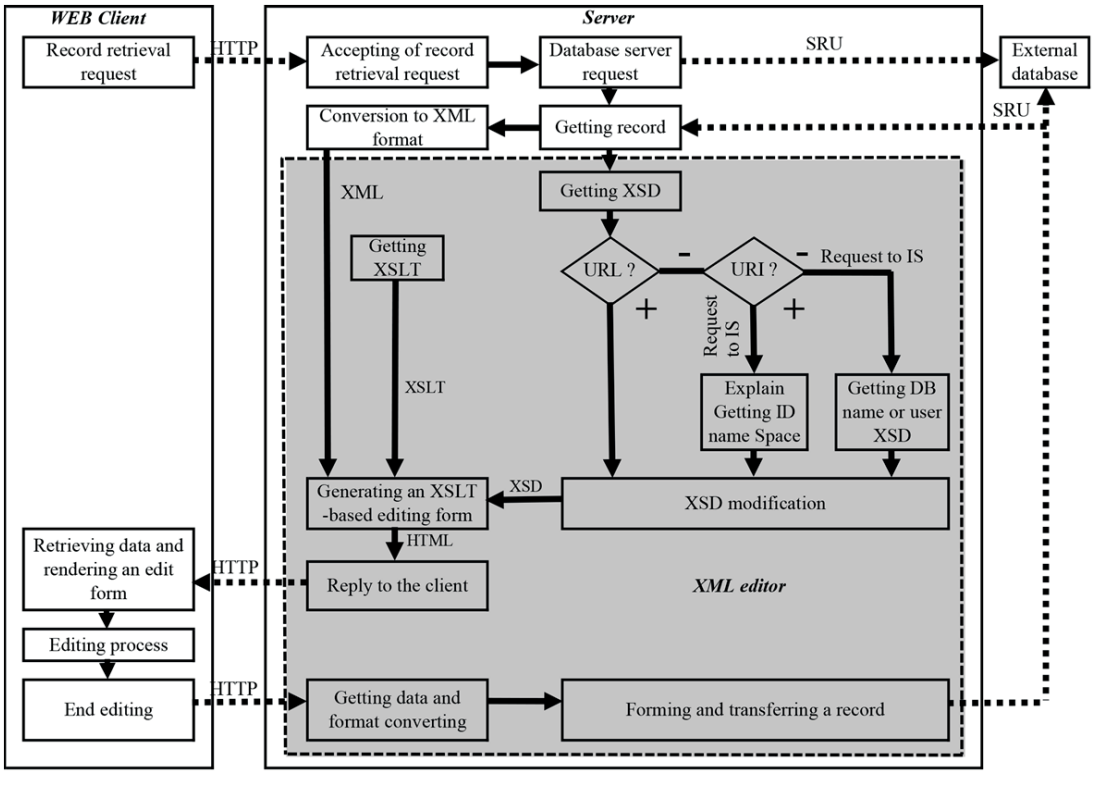

Fig. 2. Functional diagram of the operating procedure of the adaptive editor in the user-server architecture

As indicated in Fig. 2, the process of transforming XML data into an HTML form occurs by generating an XSLT-based form of editing:

Input document:
Source XML-file (*.xml);
XML Schema (*.xsd);
Extensible stylesheet XSLT (*.xsl).

Output document:
HTML-page (*.html).

The generated HTML form serves as a data entry form, i. e. is a simple low-level web-based XML editor.

The main functions of this editor are as follows:

– presentation of XML records in a browser window in a user-friendly form;

– ensuring efficient data entry and editing;

– generation of new output XML documents;

– ending XML documents to the DBMS.

The originality of this work is in the functional diagram of the XML record editor in the subsystem, which speaks about the XML record editor principle in the user-server architecture and comes in the following form:

– clients are provided with a ready-made HTML form for entering and (or) editing data. And, the form already contains all the necessary tools (java scripts) for correct data entry;

– the editing form is generated on the server using the XSLT method of transforming the modified XSD structure. First, an empty edit form (with no data) is generated, which, after the XSLT processor's operation is completed, is filled with the record data in XML format.

Also, the implication of this study is the client part, which represents a ready-made HTML form for entering and (or) editing data, which contains all the necessary tools for correct data entry, including the following:

– script to duplicate regular elements according to XSD;

– script to remove removable elements according to XSD;

– script for verifying the correctness of data input, provided that the corresponding custom template is present in the form of a regular expression in the XSD;

– script for hiding/opening of any data element in the editing form.

Fig. 3, 4 show HTML form fragments generated by the XML record editor. Information entry fields are provided next to the field names.

It should also be noted that XSD schema definitions may contain references to other XSD schemas that supplement definitions in both the current namespace (the xsd:include element) and other namespaces (the xsd:import element). Therefore, the original XSD structure needs to be modified to record additional definitions before being processed by the XSLT processor.

As noted above, the editing form is generated on the server side using the XSLT method of transforming the modified XSD structure. A blank edit form (with no data) is first generated, which is filled with the record data in XML format, after the XSLT processor has finished operation.

When generating an empty edit form, the following rules are complied with.

A window is generated indicating the data schema identifier.

Documents (XSD summary of the data element) are generated:

– a window indicating the name of the element and its position (in XPath style) in the XML record structure;

– element hiding/opening button in the editing form;

– documentation (summary), if available, indicating the language;

nested elements (for structured ones);

– element value entry field (for simple ones);

– names and data entry fields for each of the possible attributes;

– buttons for deleting (if allowed) and duplicating (if allowed) an element.

The following buttons are generated:

"Record" – to save the editing result.

"Clear" – to regenerate an empty editing form.

"Close" – to close the editing form without saving the data.

Following the requirements and analysis of the existing solutions for data integration in distributed information systems, a Web-oriented model for XML documents transformation had been developed and implemented, designed to set-up communication with a database server and fully visualize the structure of a document for its editing.

The XSLT template describes the rules for how elements from XML are transformed into HTML forms. It specifies the order, nesting of elements, screen display rules, hints, or additional information that could have been both in the XSD file and in the XML document itself.

Below (Fig. 5) is an example where an XSLT template is applied to an XML document.

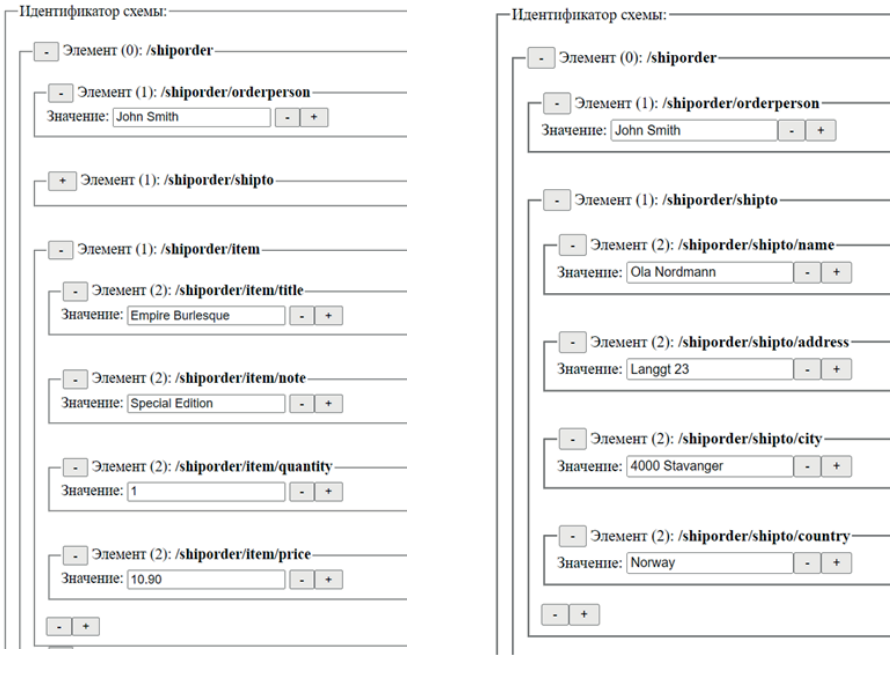

Fig. 3. An adaptive editor interface: fields names and data entry fields

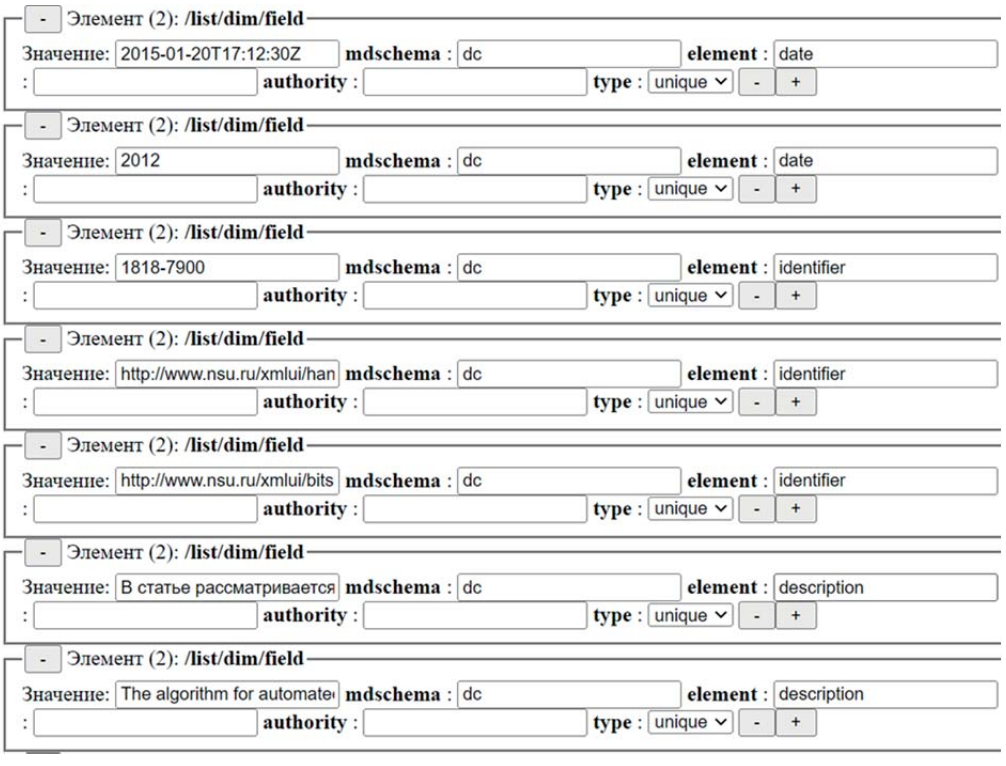

Fig. 4. An adaptive editor interface – using different types of fields

An HTML code with JavaScript fragments is obtained as a result of the conversion (Fig. 6), which is displayed in the main editor window.

An XSLT template extracts the basic attributes of elements from an XML document.

For each element, an xsl:template with a match attribute equal to the element's name is created:

<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, then the xsl:apply-templates select="*" function is added to the template with the appropriate parameters.

The correct XSLT functions and elements are added to the xsl:template so that xsl:template can generate the required elements for input, where needed.

A new XML document is created when the Record button is clicked. The HTML form is checked against the pre-built data model, and if no errors are found, a new XML document is generated.

```
<xsl:template match="/xsd:schema">
    …
    </head>
        <body>
            <br />
                <h2 class="m_h2"> XML editor</h2>
                <h3 class="m_h3"> Version 0.8</h3>
            <br />
                <form action="save.php" method="post">
                    <input type="hidden" name="nameSpace">
            <xsl:attribute name="value">
                <xsl:value-of select="@targetNamespace" />
            </xsl:attribute>
                    </input>
```

Fig. 5. Fragment of an XSLT template code

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8" />
            <title> XML editor</title>
                <link href="instr.css" type="text/css"
                rel="stylesheet" />
        <script language="javascript" type="text/javascript">
        <xsl:comment>
```

Fig. 6. Fragment of the HTML code after transformation

The main task is as follows:

1) for each element, create an xsl:template with a match attribute equal to the element's name;

2) if it is a datatype element, add it and the functions needed to create the element's value to xsl:template;

3) if it is a complex element, add the xsl:apply-templates select="*" function;

4) for each element, create an xsl:template with the match attribute equal to insert-elementName, where elementName is the element name;

5) if it is a complexType element, recursively add all of its child objects N times, where N is the value of the minOccurs attribute for the child element.

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

– straight text;

– comments;

– element attributes;

– complex element that may have a value, attributes and that must contain other elements inside;

– simple element, has only a value and attributes.

In the course of generation of data entry fields, the data type and imposed restrictions must be taken into account. In particular, the input field for elements and attributes will represent a drop-down list of values (Fig. 7) where there are the following 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>.
```

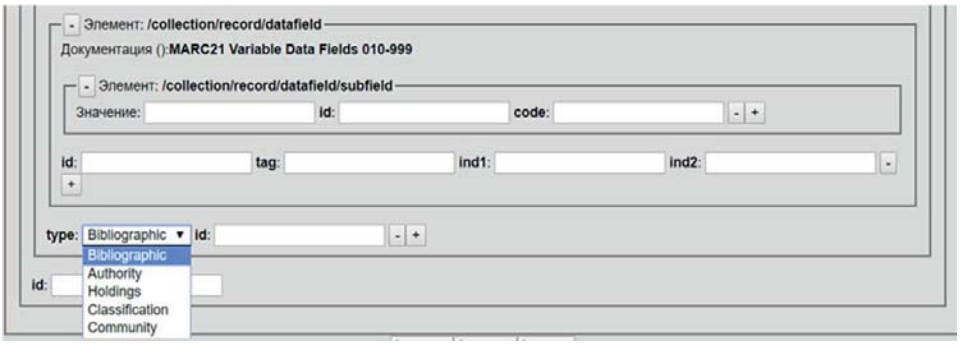

Fig. 7. An adaptive editor interface: screen of a drop-down list of values

Where the 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 in the editing form, a call to the function of verifying the compliance with the input data template is generated, 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, in turn, will generate the elements of the form:

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

The problem of recursive definitions that occurs when using references to types and names, the possibility of having XML elements with unlimited XPath length, by controlling the number of attachments and restrictions based on the current need has been solved. The entered data is automatically saved in the same database from which the record was retrieved.

5. 2. Experimental implementation and effectiveness analysis of the proposed model in the distributed information system

The sequence diagram given in Fig. 8 clearly shows the place and purpose of an XML editor in a distributed information system. The process from the moment of request (reference) and receipt of the resulting web page by the user to the system. The editor itself is located on the server side with access to databases, the levels and areas of access to which differ depending on the capabilities provided to different users.

Due to implementing the XML update model, an important criterion is the time it takes to generate the resulting HTML page on the client side. For comparison, we measured the time of generating the resulting HTML page in various situations with or without the described adaptive editor (Table 1).

Before the integration of the created data editing model, this function in the system was performed by the Explain service. It provided access through search and retrieval services to the IR-Explain-1 database. The extracted data was presented to the user in the Explain syntax, which has a special form that is not very convenient for manipulating data.

Also, the time of generating a response to the client using the XML editor in various modifications of the file (by URL, URI or by database) was measured without taking into account the time of accessing the resource to generate the XSD.

The following results were obtained.

Table 1

Formal representation of response time to the client

Time without an XML editor	Time within an XML editor
a (ms)	a+O(n) (ms)

In the above designations, a is the time it takes to enter data to the document from the incoming XML file; n is the number of elements in the entered XSD data, and O(n) is the time it takes to bypass an individual XSD document.

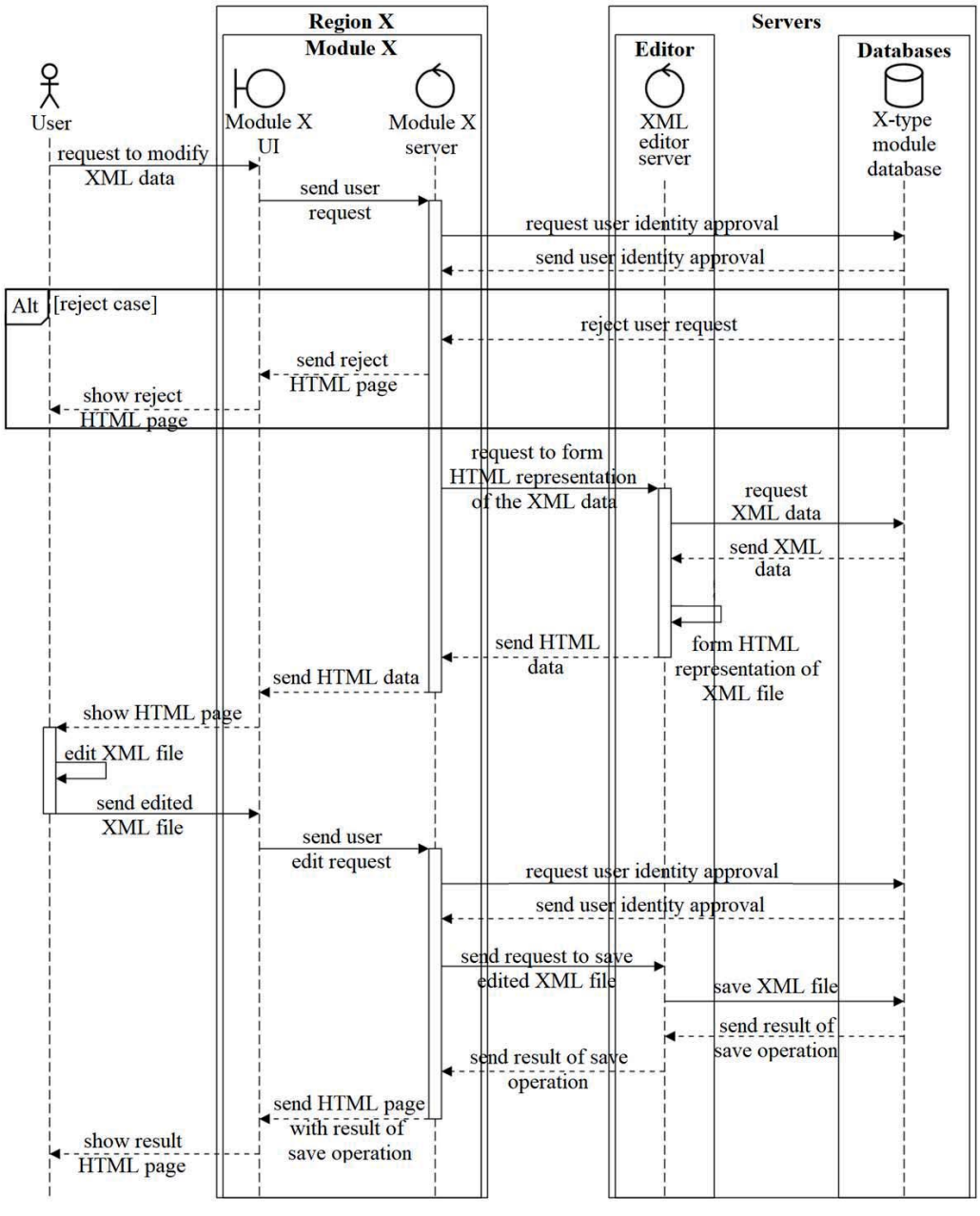

Fig. 8. The editor sequence diagram

On average, for documents present in the IS and for which measurements were taken, the following values were obtained: a=2, O(n)=6.

6. Discussion of experimental results of the development of a web interface model for editing XML data

As part of the goal of the study, a model (Fig. 1) for extracting the structure of an XML record for editing data was developed. This model generates XSLT-to-XSD transformation rules in order to obtain all possible ways to fully describe the possible structure of an extracted record (XSD). In contrast to the technology proposed by the authors [12] – a tree-like display of a document in HTML, the created model is able to display the complete structure of the file in a graphical interface, indicating the names of all fields and data entry fields (Fig. 3) with the possibility of manipulating them. In contrast to the works [7, 10], a convenient web interface is presented with all nested scripts for working with data.

In comparison with such editors available on the market as Arbortext Editor, XmlGrid.net, Oxygen XML Editor, the developed editor is able to adapt to system resources – to extract a description of the file structure from all available system resources (Fig. 2). The model (Fig. 4) implemented in this study retains all identified XML constraints after any update operation (i.e. insert, delete, or replace) performed on the XML data. The XML document is updated and its schema automatically adapts without any user or developer involvement.

The study found that there are no subsystems in the existing distributed information systems adjusting to the structure and data semantics in the form of an adaptive model for entering and editing information. An important factor in systems of such type is that the structured XML format,

after transmission over the network, makes it possible to completely preserve the original record structure, unlike other protocols (http, ftp, etc.).

Table 1 compares the quality characteristics by the time of generating a response to the client without an XML editor and within an XML editor in various file modifications (by URL, URI or database) excluding taking into account the time of accessing the XSD generation resource. The result showed that the speed of generating a response to the client is 3 times faster when using the editor.

However, a web page with a document to be edited can take a long time to form; in exceptional cases reaching several seconds per document. This fact is explained by the fact that files located on resources with a long response time are used as third-party XSD schemas. When using resources with a fast response time, no such situations arise. This may require research in the field of data caching applicable to the described adaptive scheme.

Evaluation of the research results by introducing the developed technology into a distributed information system has demonstrated the efficiency of this technology in the form of increasing the features and enhancing user-system interaction based on the XSD application database schemas using XSLT transformations. The described methodology has got rather general designated uses and can be applied to build adaptive graphical interfaces that help generate the sent HTML forms for data entry and editing.

The implementation of the above concepts ensures reliable interaction between client and server applications on various platforms when operating in heterogeneous environments.

The developed adaptive XML records graphical editor allows one to import any XML data and transform its structure efficiently and easily, while the same processing procedure makes it possible to transform the source data of any structure without any change in the program code.

A particularly critical advantage of a specialized interface for entering and editing XML data is its integration with the web environment, as well as platform independence.

For the integration between web systems using a different data format as the main (JSON or XML), the converting mechanism from one format to another is needed. The existing JSON data converters in XML do not provide an interface for validating and verifying output data for compliance with a certain structure, which can lead to their loss or incorrectness after editing during reverse transformation. We consider it advisable to conduct subsequent research in this area. The purpose of which is to create a model of the JSON-XML data converter (input – JSON, output – XSD and XML), which would take into account the correspondence of the output data of a given structure.

7. Conclusions

1. A model for extracting the structure of an XML record for non-well-formed documents was developed. All possible ways to obtain a complete description of the possible structure of the extracted record (XSD) have been realized. Due to its adaptability, this model provides screen forms for creating and editing XML documents in accordance with the selected XSD schema and is able to determine web resources and systems that use them. This model is capable of providing extended addressing for finding well-formed/non-well-formed XML documents. The created web interface fully displays the content of XML documents and has a convenient form for data manipulation.

2. An analysis of the effectiveness of the implementation of the resulting model in the distributed information system, based on XML/SOAP/SRW technologies, showed that in the process of extracting records from databases, data is processed in all possible representations:

– internal representation of data in a particular DBMS – the form in which data is stored;

– internal abstract server representation – the form in which data is processed by the server;

– the external view is the form in which the data is passed to the client.

The developed graphical interface allows any search engines to automatically exchange and read information among themselves without human intervention. Due to its adaptability, it significantly improves the search and editing of documents, thereby increasing the performance of the system and the effectiveness of its use.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Financing

The study was performed without financial support.

Data availability

Data will be made available on reasonable request.

References

1. Brahmia, Z., Hamrouni, H., Bouaziz, R. (2020). XML data manipulation in conventional and temporal XML databases: A survey. Computer Science Review, 36, 100231. doi: https://doi.org/10.1016/j.cosrev.2020.100231
2. Bajaj, A., Bick, W. (2020). The rise of NoSQL systems: Research and pedagogy. Journal of Database Management, 31 (3), 67–82. doi: https://doi.org/10.4018/JDM.2020070104
3. Jounaidi, A., Bahaj, M. (2018). Converting of an xml schema to an owl ontology using a canonical data model. Journal of Theoretical and Applied Information Technology, 96 (5), 1422–1435. URL: http://www.jatit.org/volumes/Vol96No5/24Vol96No5.pdf
4. Nikiforov, D. A., Korj, D. V., Sivakov, R. L. (2017). An Approach to the Validation of XML Documents Based on the Model Driven Architecture and the Object Constraint Language. A.P. Ershov Informatics Conference. doi: http://dx.doi.org/10.13140/RG.2.2.16542.23364

5. Tekli, G. (2021). A survey on semi-structured web data manipulations by non-expert users. Computer Science Review, 40, 100367. doi: https://doi.org/10.1016/j.cosrev.2021.100367
6. Bao, L., Yang, J., Wu, C. Q., Qi, H., Zhang, X. Cai, S. (2022). XML2HBase: Storing and querying large collections of XML documents using a NoSQL database system. Journal of Parallel and Distributed Computing, 161, 83–99. doi: https://doi.org/10.1016/j.jpdc.2021.11.003
7. Brahmia, Z., Grandi, F., Oliboni, B., Bouaziz, R. (2018). Supporting Structural Evolution of Data in Web-Based Systems via Schema Versioning in the tXSchema Framework. In Handbook of Research on Contemporary Perspectives on Web-Based Systems, 271–307. doi: https://doi.org/10.4018/978-1-5225-5384-7.ch013
8. Brahmia, Z., Hamrouni, H., Bouaziz, R. (2022). TempoX: A disciplined approach for data management in multi-temporal and multi-schema-version XML databases. Journal of King Saud University - Computer and Information Sciences, 34 (1), 1472–1488. doi: https://doi.org/10.1016/j.jksuci.2019.08.009
9. Engelfriet, J., Hoogeboom, H. J., Samwel, B. (2020). XML navigation and transformation by tree-walking automata and transducers with visible and invisible pebbles. Theoretical Computer Science, 850, 40–97. doi: https://doi.org/10.1016/j.tcs.2020.10.030
10. Akazawa, G., Matsubara, N., Suzuki, N. (2022). An Algorithm for Transforming Property Path Query Based on Shape Expression Schema Update. SN Computer Science, 3, 196. doi: https://doi.org/10.1007/s42979-022-01086-0
11. Mahmood, A. T., Kamil Naser, R., Khalil Abd, S. (2022). Privacy protection based distributed clustering with deep learning algorithm for distributed data mining. Eastern-European Journal of Enterprise Technologies, 4 (9 (118)), 48–58. doi: https://doi.org/10.15587/1729-4061.2022.263692
12. Fan, C., Li, Z. (2019). Research on Addressing Method in XML File Based on XPointer. Advances in Graphic Communication, Printing and Packaging, 384–389. doi: https://doi.org/10.1007/978-981-13-3663-8_52