Document transformations for data processing in information systems [1]

Łukasz Budnik[2], **Henryk Krawczyk**[2]

Abstract: The paper presents XML-based approach to automating user's document lifecycle transformations. It discusses a user case of Endoscopy Recommender System. The ERS system uses self-developed XSD to Java, Java to XML transformations that allow data acquisition and data storing processes to be fully automated. The implemented XML data binding approach provides validation of basic types, lists and other objects. ERS uses IBM DB2 pureXML engine as an efficient XML storage. In addition, ERS utilizes the IBM DB2 XSR repository to provide XML validation and to keep data consistent.

Keywords: document transformations, document lifecycle, database engines, web-based systems, programming platforms, execution environments

1. Introduction

All information processing systems base on users' documents. Throughout all variety of both obstacle and modern systems, these documents conform always to the very same lifecycle. There are three basic lifecycle states that can be defined as:

- user's document state — interacting phase, in which user actually operates on the document and interacts with it
- business object state — processing phase, in which the document as system's business object is processed (the business logic)
- database record state — storing phase, in which the values of business object are saved in persistent storage

The flow through the lifecycle states is given in Fig. 1. The pattern, shown in Fig. 1 can be implemented in different ways. However, designing and implementing efficient and automated transformations is always a challenge. Many systems, especially legacy systems, suffer from severe difficulties when transforming user's documents. Not all transformations fit precisely, most of them is not processed automatically. Information processing system can benefit greatly from carefully designed, advanced, and automated transformations mechanism. On the other

[1] The paper is sponsored by the Polish Ministry of Science and Higher Education grant No. N519 022 32/2949

[2] Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, ul. Gabriela Narutowicza 11/12, 80-952 Gdańsk, Poland
e-mail: `lukasz.budnik@vega.eti.pg.gda.pl, hkrawk@eti.pg.gda.pl`

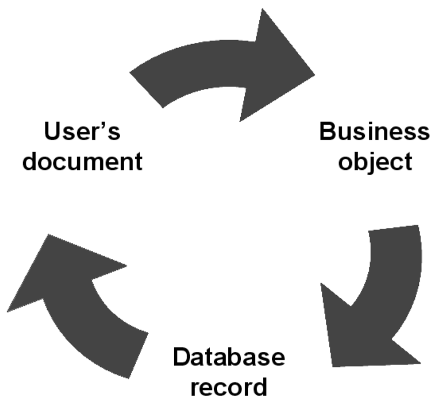

Figure 1. User's document lifecycle

hand, poorly designed mechanism can be a serious threat to the stability and reliability of the whole system.

XML has been chosen as a base of the new system. XML is a perfect solution for exchanging data. Recently presented research results show that the number of XML data has grown 5 times during 3 years, from 2004 to 2006 (see Borre, 2006). In modern information systems more and more business objects are made available for external users. For example, many information systems come with Web Services modules that allows remote access to specific data in XML format. This results in an increasing number of systems that use XML data binding for their business objects. XML data binding technique provides a direct way to use XML documents in various applications without knowing the actual structure of XML documents and given programming language XML API itself. Although there are some limitations, this approach can be successfully used in simple applications that use fixed-structure XML for data exchange.

The popularity of XML is also reflected in trends of developing modern databases. IBM came in 2006 with IBM DB2 9 pureXML (see Nicola, 2006 and IBM, 2006) — first hybrid database. Also, Oracle and MS SQL provide XML data type. Some of the open-source databases begin to implement basic XML support, for example, MySQL 5.1 provides functions for extracting and updating XML documents.

XML allows to distinguish and separate structure, content, and style (see W3C, 2006). XML's features are as follows:

- productivity — concentration on the content of the document, highly optimized syntax, user can create own markups and structures
- usability — readability for machines (ordinary text file)
- data sharing — integration, localization and extraction

- portability — independence of hardware and software

To support XML technology, WC3 developed two standards that are strongly linked with it. These are: XML Schema (XSD) (see W3C, 2004) for defining XML structure, and XSLT (see W3C, 1999) for expressing style sheets, styling and transformations. What is more, both XSD and XSLT are defined as XML documents themselves.

XML Schema is the ultimate solution for describing XML documents. XSD defines a set of basic data types (these include numbers, strings, dates, URLs, etc.), and provides means of defining new ones: complex data types (elements containing other elements), and simple types, derived directly from basic types, with various restrictions added (e.g., string's pattern, minimum/maximum length). With XSLT one can transform XML documents into another XML or visualize content with complex styling operations. XSLT is widely used for generating XHTML web pages from XML data.

In the paper we show how to implement XML-based user's document lifecycle in a real, medical systems we consider. The system's name is Endoscopy Recommender System. The system's business client is Gastrointestinal Endoscopy Clinic at the Medical University of Gdansk. ERS's task is to assist physicians in their work. Currently used version of the system — ERS 2005 (see Krawczyk, 2006) is not able to cope with dynamics of the changes taking place in medical procedures and requirements set by physicians.

To achieve such capabilities we propose a distributed web-based system that aims at efficient acquisition and validation of medical data; it should store, process, and output the data in visually attractive and readable way.

2. Verified user's document lifecycle

The presented earlier user's document lifecycle (see Fig. 1) is only conceptual and needs to be evaluated in detail. With all modern powerful enterprise solutions at our side, we decided to simplify to the maximum the user's document lifecycle. At first, we simplified user's document states by introducing only two states, that are as follows:

- business object with implemented business logic
- serialized object as an XML document — used for both presentation and storage

Also, the proposed solution includes propositions of automated mappings between these two forms.

For describing XML medical documents we used XML Schema. With all the information provided by schema, one can use it for generating business objects (Java or any other OOP language) and for data validation (based on XSD restrictions).

Our idea was to build auxiliary stand-alone application that could read XML Schema and transform it into Java classes that could be used by the ERS system. Such application must consists of two modules:

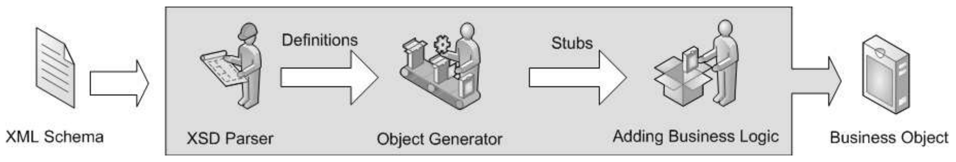

Figure 2. Idea of the XML Schema to business object

- XSD parser — a module that traverses XML Schema and extracts all types definitions
- Java objects generator — a module that, basing on parsed XML Schema, generates classes with proper fields, dedicated getters/setters, and serialize/deserialize methods

At the end of the whole process, a programmer can implement business methods for calling other objects, sending messages, etc. The idea of XML Schema to business object is shown in Fig. 2. As a part of our approach, an example data acquisition layer was proposed. New fully automated data acquisition layer was meant to implement web forms to business objects mapping mechanism. The mechanism would allow business objects to be automatically instantiated with all their fields populated with the corresponding web form fields' values. The suitable mapper should work in two ways:

- XForm-to-BusinessObject — business object must be automatically created and its fields populated after the form is submitted
- BusinessObject-to-XForm — web form fields must be populated based on bound business object — for example, used during data editing

As a last stage of our XML-oriented approach, we propose an efficient XML data storing mechanism. XML can define highly flexible structure, when compared to relational model and storing XML documents in relational tables is a step back (see Nicola, 2005). IBM DB2 pureXML has revolutionized support for XML data in modern databases. XML documents are stored in dedicated XML column type that preserves document's hierarchy and allows to traverse its contents with XPath queries. IBM DB2 pureXML engine comes with three new query types: XQuery (see Chamberlin, 2002, W3C, 2007), SQL with embedded XQuery, and XQuery with embedded SQL.

All mentioned earlier proposals led us to defining new, fully automated user's document lifecycle. The revamped and detailed lifecycle including XForm acquisition, business object serialization/deserialization, and XML storage is presented in Fig. 3. The basic user's document states are placed in boxes.

3. Architecture proposition

We describe approach to a flexible framework for web front-end with a high abstraction level that can deliver various business features. Our goal was to dispatch

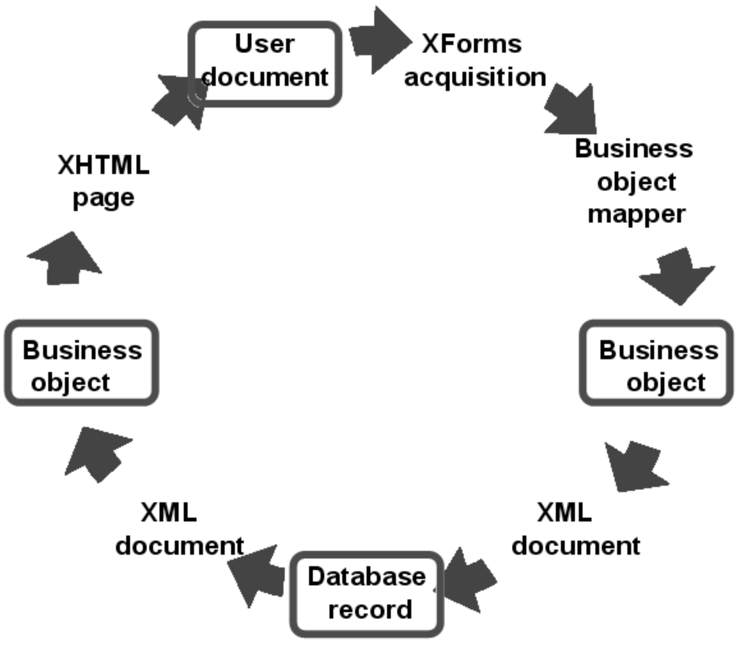

Figure 3. Revamped user's document life cycle

transparently user's requests without interfering in both input and output parameters.

The chosen architecture in proposed solution is based on JEE technology. All enterprise applications consist of two main modules, that in JEE naming convention are called:

- WAR — web application, front-end, responsible for authorization, presentation, request dispatching, and look and feel — the dispatching domain
- EAR — enterprise application, back-end, delivers business solutions — the business domain

For the final user, single business solution is delivered by a web page. Web pages are grouped by their functionality in web modules. For example, *Add* web module may contain many logical web pages performing additions like: *AddPatient*, *AddExamination*, etc.

The only link between WAR and EAR applications is the JNDI directory, in which EAR registers its EJB modules. Fig. 4 presents the link between WAR and EAR problem domains, and the actual localization of user's document states.

The idea of separating business and presentation code bases on the direct use of XML. User's requests are pre-processed on a web container and later on, passed to

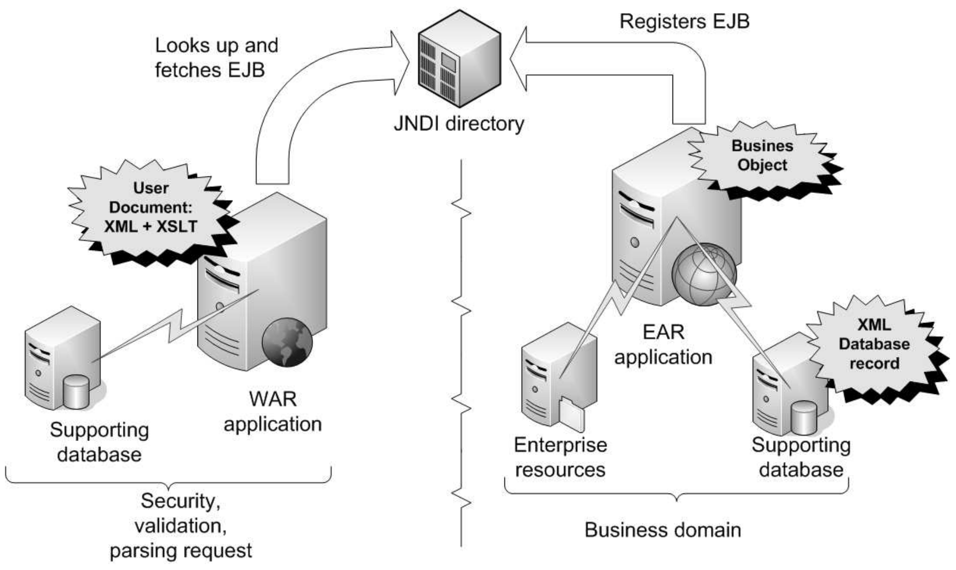

Figure 4. The link between WAR and EAR applications

the enterprise application on a remote application server. The dispatcher locates the remote EJB module, calls its business method, and waits for its XML response.

When the response is ready, the XML document is sent back to the web application and according to the configuration and parameters, a proper output is generated. Most frequently, the response is later transformed with XSLT style sheets, and the final response is sent to the user.

4. User case study

Below we show how the document lifecycle is implemented in an XML-oriented system. We consider the ERS 2007 system, where the presented in Fig. 3 transformations were used for data acquisition related to patients' personal details and their health information.

The implemented XSD Parser was meant to be a simple parser that works only with a small subset of the whole XML Schema standard. The basic concepts of both XSD parser and Java business classes generator were:

- mapping complexTypes to classes — a complexType can contain other complexType
- mapping simpleTypes to Java native types with restrictions checked before every set operation, also null values assertions
- mapping maxOccurs occurrence indicator to dynamic lists
- proper parsing and formatting XML Schema/Java types — for example dates

- generating serialize and deserialize methods — two way XML serializing

In the proposed solution the class generator based on read XML Schema definitions and their elements' restrictions appends validation code to all setter methods. Combined with the object-oriented encapsulation paradigm — modifications of private fields are possible only by calling public setter methods — fields of given business object will always have valid values. Although implemented XSD parser supports only a small subset of XSD, the business object generator takes full advantage of supported definitions. In addition, business object generator creates two setter methods for each field. First is called native setter that expects Java object and provides validation. The other is a string setter that expects a string, converts passed values to Java objects and calls the first, native one; it was introduced to facilitate deserializing of XML documents (XML document is a string).

Given snippet of XML Schema as presented:

```
<xs:element name="patient" type="patientType"/>
<xs:element name="address" type="addressType"/>
<xs:element name="phone" type="phoneType"/>
<xs:simpleType name="phoneType">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]{3}(\-[0-9]{3}){2}"/>
</xs:restriction>
</simpleType>
<xs:complexType name="addressType">
  ...
</xs:complexType>
<xs:complexType name="patientType">
  <xs:element ref="address" />
  <xs:element ref="phone" minOccurs="1" maxOccurs="3"/>
  ...
</xs:complexType>
```

The snippet of automatically generated Java stub would look like:

```
public class Patient extends AbstractBusinessObject {
  private Address address;
  private List<String> phone;
  // native setter, provides validation
  public void setPhone(List<String> phone) {
    if (phone.size() < 1) {
      // throws exception;
    }
    if (phone.size() > 3) {
      // throws exception;
    }
    for (String i : phone) {
      if (!Pattern.compile("[0-9]{3}(\-[0-9]{3}){2}").
        matcher(i).matches()) {
        // throws exception;
      }
    }
    this.phone = phone;
  }
  // string setter, converts string representation to Java object
  // calls native setter public
  void setAddressString(String address) {
    Address obj = new Address();
    obj.init(address);
    setAddress(obj);
  }
}
```

The implemented XForm mechanism allows business objects binding and automated fields mapping. XForm can map any number of fields. For example, after examination physician completes examination file with the description and diagnoses, there is no need in displaying once again all fields. In addition, a single XForm can define any number of classes when complex objects need to be acquired; for example, Patient and its Address objects' values can be acquired in a single XForm. When submitted, XForm detects "class" attribute and uses Java Reflection API to automatically map web form's fields to corresponding business object's fields. Furthermore, XForm is not only a static HTML component, it provides interactive validators for all fields types. Interactive validators increase user-friendliness. Based on AJAX technology, they validate fields behind the scene each time the value is changed; the user is almost instantly (AJAX is asynchronous) notified whether the entered value is correct or not. The following example shows basic, one field XForm that defines a required text field with a callback function that will be automatically invoked after successful validation.

```
<form id="Patient">
<group id="dane" class="Patient">
  <caption>Patient's personal details</caption>
  <description>
  Please fill in all basic patient's personal details</description>
  <text id="ssn" required="1" callback="parseSSN">
    <desc>SSN</desc>
    <help>Please fill in the Social Security Number.</help>
    </text>
</group>
</form>
```

In addition, ERS 2007 XForm defines powerful XFormDataSource mechanism that allows to dynamically fetch records from DB or from request scope and create web form's fields and their values. The statements for XFormDataSource are expressed in two ways: XQuery, simple query with hard-coded conditions, for example, list of all working physicians, and XQuery with embedded SQL when dynamic markers are needed, for example, list of examination types according to given request scope parameter. A more sophisticated example of retrieving diagnoses according to given examination type is presented next, as follows.

```
<datasource>
  <type results="set">sql</type>
  <query><![CDATA[XQUERY
declare namespace tns="http://www.dataweaver.org/diagnosis";
for $x in db2-fn:sqlquery('SELECT DIAGNOSISDESCRIPTION FROM
ERS_DIAGNOSES WHERE idRange = (SELECT DISTINCT idRange FROM
ERS_EXAMINATION_TYPES WHERE
idExaminationType = :examinationType)')/tns:diagnosis
order by $x/tns:name/text() ascending
return (
  <field><value>{string($x/@idDiagnosis)}</value>
  <desc>{$x/tns:name/text()}</desc></field>
)]]></query>
  <markers>
    <variable scope="request" name="examinationType"/>
  </markers>
</datasource>
```

5. Deployment environments

As a deployment environment for both WAR and EAR, an application server must had been chosen. There are many application servers available on the market, but only few count. The most feature-rich, well documented, commonly used and open-source application server is JBoss (see JBoss, 2007). JBoss extends Apache Tomcat web container and adds enterprise support. JBoss has also one distinguishing feature — it comes with bundled lightweight and efficient SQL database — Hypersonic SQL (HSQLDB) (see HSQLDB, 2006).

WAR application needs its own supporting data source containing information about deployed, available and ready to use remote modules. JBoss, with its bundled HSQL database, is an ideal choice. It has everything one may need when deploying simple front-end application. Thanks to bundled HSQL database developers and people responsible for deployment do not have to worry about installing and managing additional database engine — HSQL is already configured and starts automatically with JBoss.

The chosen solution for the new database was a hybrid approach. Hybrid approach takes full advantage of both relational model and XML data type. In ERS 2007 every form of the user's document has always full information of its objects inter-relationships. Although serialized document has all foreign keys information and it is possible to use them in XQuery joins, we decided to introduce relational foreign keys to elevate the overall performance of the whole database. Thanks to the use of relational engine all joins and sub-queries are executed very fast. Moreover, relational foreign keys have additional features for keeping the database consistent — with foreign keys constraints, and on update and on delete cascade actions.

Apart from standard primary keys with auto-generated identities, indexes, unique constraints, and foreign keys, new XML elements (see Nicola, 2005) in ERS 2007 database are:

- data storing in hierarchical XML structures
- using XML Schema Repository (XSR) and XML validation in DML statements
- using generated keys with XMLPATTERN to speed up XQuery statements

The primary and foreign keys allow to traverse the whole structure from every possible logical point. Only specific filter conditions are expressed in XQuery.

In our solution, the very same XML Schemas that were used in the business objects generation process were deployed in the IBM DB2 XSR repository. Thus the whole ERS 2007 database will always provide coherent results even when modifications were performed by a different application.

To deploy XML Schema in IBM DB2 XSR repository one must issue following command:

```
REGISTER XMLSCHEMA 'http://www.dataweaver.org/patient'
  FROM 'file:///H:/MSc/DataWeaver/src/xsd/ea/Patient.xsd'
  AS PATIENT COMPLETE
```

After successful registration, XML validation is possible in any DML statement. For the registered above XML Schema, sample JDBC stub of INSERT statement would look like:

```
INSERT INTO ERS_PATIENTS (PATIENTINFO) VALUES (
  XMLVALIDATE(XMLPARSE(DOCUMENT cast(? as CLOB))
  ACCORDING TO XMLSCHEMA ID PATIENT)
)
```

6. Comparison of ERS 2005 and 2007 frameworks

ERS 2005 is a fully object-oriented system written in PHP5. ERS 2005 has specialized engine, due to which it is possible to obtain modular and easy to modify application, which was a highly desirable effect. However, there are some very serious problems in ERS 2005. ERS 2005 drawbacks concentrate around inefficient database project from year 2001 and enterprise capabilities of PHP5 technology itself.

The previous version — ERS 2005 and the current version — ERS 2007 are compared to stress user's document lifecycles and their transformations. The comparison is presented in Table 1.

Table 1. ERS 2005 and ERS 2007 document's transformations and their life cycle comparison

Feature	ERS 2005	ERS 2007
Internal data storing and representation	XML and XSLT	XML and XSLT
DBMS engine	MySQL5, inefficient relational database from year 2001	IBM DB2 pureXML, efficient, new hybrid database
Business objects	No business objects, all fetched records stored as associative arrays	Generated automatically based on XML Schemas with appended validation code and XML serialize/deserialize methods
Data access	No DAO classes	Simplified to the maximum XML DAO classes with IBM DB2 pureXML validation mechanism
Acquisition layer	No automatic mapping supported, regular expression validation possible, API for creating dynamic fields and filling their values available	Fully automated two way mapping: BusinessObject–XForm, advanced interactive validation based on XSD restrictions, XFormDataSource mechanism for generating dynamic fields from DB and request parameters

The ERS 2007's framework takes full advantage and uses to the maximum the XML technology. The XML technology is used in every phase of user's document lifecycle:

- processing — based on XML Schema definitions ready-to-use business objects are automatically generated; setters have validation code appended with proper converting operations between XSD and Java types, also serializing/deserializing XML methods are created
- storing — business objects can serialize themselves to XML documents that are later on stored in hybrid XML database; in addition, XML Schemas used during business objects generation are deployed in IBM DB2 pureXML XSR repository, thanks to registering XML Schemas in XSR we benefited from validation of XML-based DML statements and keeping records consistent
- interacting — XML representation of a document is visualized with XSLT

7. Conclusion

In order to deploy the ERS 2007 system, all Gastrointestinal Endoscopy Clinic's data must be migrated. This process must apply to over 40 000 patients and 60 000 examinations (not including other 13 tables).

To migrate old data we can use generated business objects from the ERS 2007 system. Based on fetched records proper business objects can be created and their fields populated. After serialization, XML records can be imported to IBM DB2.

Our main goal is to propose a framework capable of automated user's document transformations with the use of the most powerful, enterprise solutions. Due to greater flexibility of system's distributed architecture and design, simplifying and automating user's document transformations we managed to reduce the time and cost of development and maintenance (for example, extending existing business objects, changing XML database structure). In ERS case, previous implementation phase of business logic took 6 months, and the present one only 3 months.

Our experience shows that it is worth spending additional time in learning new innovatory solutions and later on taking full advantage of their features. It is also profitable to create auxiliary applications to automate certain software engineering lifecycle phases.

References

van den Borre, S. (2006) *DB2 9 & XML.*

Budnik, Ł. (2007) *Distributed system for gathering and analysing XML data*, Master's Thesis, Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology.

Chamberlin, D. (2002) *XQuery: An XML query language.*

HSQLDB (2006) *Lightweight 100% Java SQL Database Engine.*

IBM (2006) *IBM DB2 Database for Linux, UNIX, and Windows Information Center.*

JBoss (2007) *JBoss Application Server.*

Krawczyk, H., Dusza, K., Budnik, Ł. and Byczkowski, Ł. (2006) *Multidimensional Legacy Aspects of Modernizing Web Based Systems.*

Nicola, M. and van der Linden, B. (2005) *Native XML Support in DB2 Universal Database.*

W3C (1999) *The Extensible Stylesheet Language Family (XSL).*

W3C (2004) *XML Schema.*

W3C (2006) *Extensible Markup Language (XML).*

W3C (2007) *XML Query (XQuery).*