SRU/W service for CRIS UNS system

Valentin Penca*, Siniša Nikolić *, Dragan Ivanović *
* University of Novi Sad/Faculty of Technical Sciences/Department of Computing and Automatics, Novi Sad, Serbia
{valentin_penca,sinisa_nikolic, chenejac}@uns.ac.rs

***Abstract*—. This paper describes an independent, modular software component that enables search and retrieval of scientific research data from CRIS system in accordance to SRU/W standard. The component is implemented as an extension of existing CRIS UNS system of University of Novi Sad.**

I. INTRODUCTION

The development of science has been accelerated with the appearance of information systems for managing data. Standardization of such systems is very important. System that contains information on publications, events, research products, researchers and research institutions would have to be based on generally accepted standards. An appropriate example of such system is the Current Research Information System (CRIS) [1] that is based on the Common European Research Information Format (CERIF) [2] standard. CERIF standard represents the physical data model [3] and provides the exchange of XML messages between communicating systems [4]. Today, any respectable scientific research institutions should use some form of CRIS system.

CRIS can include various scientific research data from digital libraries, institutional repositories (IR) and other research information systems. It is evidently that CRIS systems store diverse types of scientific data, so it is necessary to provide an efficient search mechanism which should be based on a standard. The standard should provide independence of search process from scientific research data models.

The necessity for a standardization of the search is described in a series of papers. The most widely used standard in the area of search of digital contents is certainly Z39.50 [5]. One of the many examples of using Z39.50 standard is described in [6], where it is used for searching and connecting of the Iranian libraries.

However Z39.50 has certain drawbacks that new generation standard SRU is trying to overcome. SRU standard is trying to keep functionality defined with Z39.50 and to provide its implementation using currently available Internet technologies. One of the main advantages of SRU compare to Z39.50 is possibility of SRU to exchange XML messages which is not allowed by Z39.50.

Papers such as [7] and [8] confirm the use of the standard SRU/W. Zarić has described a client application that has an ability to connect to remote servers by Z39.50 or SRU/W protocols and to simultaneously search data of remote servers.

Apache OpenOffice started a new Bibliographic project (OooBib) [9]. The bibliographic project will design and build an easy to use and comprehensive bibliographic facility within OpenOffice. It will be easy to use for the casual user, but will meet all the requirements of the professional and academic writer. The new bibliographic facility will utilise the latest open standards and will make the fullest use of emerging XML, XSLT and SRU/W technologies.

SRU (Search and Retrieve via URL) 2.0 has been approved as a standard by the Organization for the Advancement of Structured Information Standards (OASIS) [10].

The paper [11] states that in the CRIS systems the search functionality is often neglected, which certainly reduces their usefulness. CRIS systems contain a large amount of data that can be interpreted differently in individual CRIS systems. The problem of simultaneously information searching in several such systems is clearly reviled. EuroCRIS proposes an introduction of CERIF as an uniform standard to overcome the problem of data search. It is emphasized that the most effective method to search CRIS system should be based on metadata.

In [12] a list of records for CRIS systems is given, for which is necessary to implement the functionality of the search. The CRIS systems based on the CERIF standard should allow search of: researchers, organizational units (institutions), projects, publications, products, patents, equipment, events (conferences, workshops etc) and sponsors.

II. CRIS UNS

In year 2009, at the Faculty of Sciences and the Faculty of Technical Sciences a development of information system for managing scientific research data from University of Novi Sad (CRIS UNS) [13] was started. The first phase of development of this system was modeling, and data entry of published scientific results. Testing and verification on the results of researcher from Faculty of Sciences was performed. At the beginning of 2013, a total number of records was over 73000.

The system for search of scientific research data is integrated within the existing system for management of scientific research data. Integration of these systems is achieved by modifying the existing and adding new components. The motivation for this research was to:

- provide public access and search for data from institutions/organizations, researchers and published scientific results within the University of Novi Sad.

- use common search standards to make precondition for interoperability with similar systems that contain scientific research data.

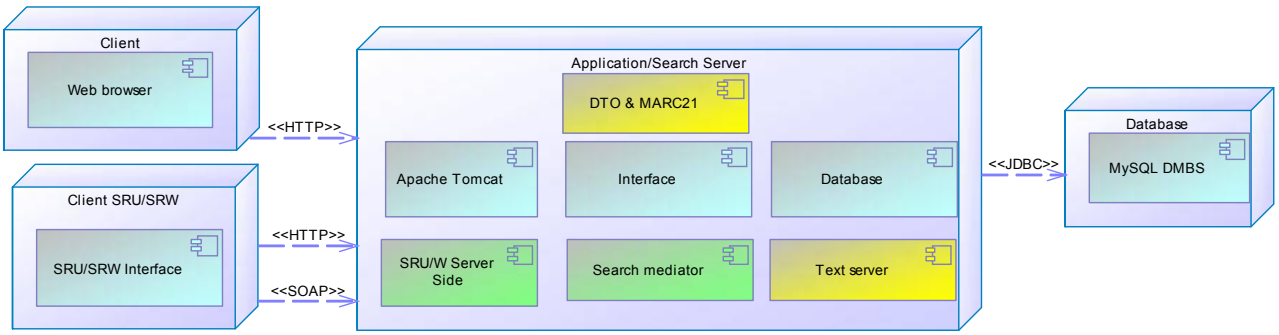

Figure 1 - CRIS UNS Architecture

Figure 1 shows the architecture of the system using the UML deployment diagram. Yellow color is used to show the modified components, while the green colored components are those added for the purpose of the search system

Client SRU/SRW - SRU/W Interface: Client-side application is an external application (usually part of another system) which implements the client side of the SRU/W protocol version 2.0. Support of context sets CQL version 1.2 [14] and the DC version 1.1 [15] is the minimum requirement for client side applications. Search of the all available records of the system is possible if the client side application supports CRIS profile [16]. The communication between client applications and server-side applications is done via HTTP or SOAP.

SRU/W Server Side: This component executes the server side of SRU/W protocol version 2.0. The *Search/Retrieve*, *Scan* and *Explain* services are implemented. Server side of SRU/W accepts e.g. Search/Retrieve queries via SOAP. Afterwards, SRU/W Server Side component invokes Search Mediator component, which will process the CQL query. SRU/W server side forwards the search results provided by the Search Mediator to the appropriate SRU/W client.

Search Mediator: This component allows to search the system by using the CQL query language. XML description of the context sets and specification of the CQL language are used for verification of the CQL query (correct syntax and semantics of a query). The system uses XML representation and JAXB (Java Architecture for XML Binding) [18] library to make object form from the context sets. CQL query is transformed into a query language of Text Server component and then the transformed query is executed in Text Server. Search Mediator component accepts and process user's CQL query from SRU/W Server Side and Interface components. Afterwards, search results are converted in accordance with the component which has initiated the call of Search Mediator component. Search Mediator component use converters that are defined in DTO & MARC21 components.

Text server: Text Server is a component based on Apache Lucene [19] library for text searching and indexing of scientific research data. Component defines searchable indexes for scientific research data from Database.

DTO & MARC 21: DTO & MARC 21 is a component which provides conversions between DTO objects and object representations of a MARC 21 [17] record. DTO (Data Transfer Object) are objects that are used to transport data between the application components. During the search, the Mediator component uses component DTO & MARC 21 for conversion of results from MARC 21 object representations in correspondent DTO objects or XML representations (MARC XML [20], Dublin Core Extended XML [21]). If the search is initiated by the Interface or SRU/W client, component DTO & MARC 21 will not start loading content of DTO objects from the database, yet it will only take their existing textual representations from Apache Lucene. Therefore, performance of the system is improved by avoiding a completely loading of records stored in the database.

III. SRU/W STANDARD

SRU/W search is based on indexes that describe different search resources. Unambiguous of search semantics and syntax, the search query is defined by CQL (Contextual Query Language) [22] and with context sets that are organized in SRU/W profiles. CQL is a common query language for searching resources where the context sets are concepts which define allowed entities in a CQL query.

SRU standard has two different implementations in which the first one can search and retrieve data by sending messages via HTTP GET and POST methods (SRU) and the other one is using the SOAP protocol (SRW) for the exchange of messages. Basic difference between SRU and SRW version is in a manner of sending messages [23]. SRW version of the protocol packaged messages in a SOAP Envelope element, while the SRU protocol version defines the message in the principle of parameter/value pairs where the parameter/value are included in URL address. Another difference between the two versions is that the SRU protocol is using only the HTTP protocol for transmission of messages, while SRW besides the HTTP protocol can use the SSH (Secure Shell) and the SMTP (Simple Mail Transfer Protocol) protocol.

Unlike the Z39.50 standard which defines 11 different services, the SRU standard defines three services (operations), since it was observed and concluded that only a certain number of services defined by the Z39.50 standard were actually used in practice. Services e defined by SRU standards are:

- *SearchRetrieve*. - service that is responsible for search and retrieval of data, where client sends *SearchRetrieveRequest* and gets *SearchRetrieveResponse* message as an answer from Server.
- *Scan* - service that allows the client to get all values from particular index. Also if supportable by the server, one more optional feature is possible. For each value of the index the number of hits obtained during the search of that index can be given. Messages that are used in communication are the *scanRequest* and *scanResponse*.
- Explain - service that allows client to send *explainReqest* in a manner to find out information about standard details which are supported by the server in a form of *scanResponse* message.

In CRIS UNS, only transport mechanism that involves use of SOAP Protocol is implemented.

IV. JAXWS

Java API for XML Web Services (JAX-WS) [24] is one of the sets of Java technologies used to develop Web services. JAX-WS is a new programming model that simplifies application development through support of a standard, annotation-based model to develop Web Service applications and clients. JAX-WS belongs to what Sun Microsystems calls the "core Web services" group. Like most of the core groups, JAX-WS is typically used in conjunction with other technologies. Those other technologies may also come from the core Web services group (JAXB, for example). JAX-WS represents remote procedure calls or messages using XML-based protocols such as SOAP, but hides SOAP's innate complexity behind a Java-based API. Developers use this API to define methods, then code one or more classes to implement those methods and leave the communication details to the underlying JAX-WS API.

JAX-WS technology has been selected for the development of web services in the CRIS UNS for the following reasons:

- JAX-WS is one of the leading technologies for development of web services based on the Java programming language
- To follow a good practice in the development of CRIS UNS on open source technologies based on the Java programming language

There are two approaches for developing web services by using JAX-WS technology:

- Developing a JAX-WS Web service from a JavaBean (bottom-up development). When developing a JAX-WS Web service starting from JavaBeans, a bean that already exists can used to enable the implementation for JAX-WS Web services. The use of annotations simplifies the enabling of a bean for Web services. It is not required to develop a WSDL file because the use of annotations can provide all WSDL information necessary to configure the service endpoint or the client.
- Another approach is to create a JAX-WS Web service, but now with an existing WSDL file using JavaBeans (top-down development). This WSDL document could be obtained from another developer, a system architect, a UDDI registry, or you could write it yourself.

In this paper, top-down development is chosen for creating JAX-WS web service. WSDL is a XML document that is used for describing web service elements, operations and structures that are used in communication. WSDL document is consisted of seven elements: `types`, `message`, `operation`, `portType`, `binding`, `port`, `service`.

Element `types` defined in WSDL specification is used to describe data types or structures that are used in message exchange process. Since, the document/literal wrapped pattern of the message is selected, each message is represented by the corresponding XML element specified within the correspondent XML schema. Schema defines six elements, for every SRU/W service (operation) two elements respectively, one for request message and another for response message. Therefore, the following messages are defined:

- `searchRetrieveRequest`
- `searchRetrieveResponse`,
- `scanRequest`
- `scanResponse`
- `explainRequest`
- `explainResponse`.

Element `binding` is used to define a protocol that will be used for message exchanging and for defining format and message encoding. For `searchRetrieveOperation` operation, whose binding element is shown in Listing 1, the HTTP protocol was selected to transport the SOAP messages.

```
<binding name="SRW-SoapBinding"type="SRWPort">
<soap:binding style="document"
transport="http://schemas.xmlsoap/soap/http"/>
<operation name="SearchRetrieveOperation">
<soap:operationsoapAction=""style="document"/>
<input> <soap:body use="literal"/> </input>
<output> <soap:body use="literal"/> </output>
</operation>
...
</binding>
```

Listing 1 - `SearchRetrieveOperation` binding

Element `portType` is consistsed from set of operations that are represented with an element `operation`, whereby for each operation are defined the input and the output parameters. On Listing 2 is presented a part of XML document that describes operation supported by SRU/W standard.

```
<portType name="SRWPort">
<operation name="SearchRetrieveOperation">
<inputmessage="SearchRetrieveRequestMessage"/>
<outputmessage="SearchRetrieveResponseMessage"
/> </operation> <operation
name="ScanOperation"> <input
message="ScanRequestMessage"/> <output
message="ScanResponseMessage"/> </operation>
<operation name="ExplainOperation"> <input
message="ExplainRequestMessage"/> <output
message="ExplainResponseMessage"/>
</operation>
```

Listing 2 - `portType` operation

Element service defines the physical location of the web service. On Listing 3 is shown an example of the web service definition that is installed on the local computer.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions .....name="SRW"> ...
<service name="SRWSampleService">
<port name="SRW" binding="SRW-SoapBinding">
<soap:addresslocation="http://localhost:8080/"
/> </port> <port name="ExplainSOAP"
binding="Explain-SoapBinding">
<soap:addresslocation="http://localhost:8080/"
/> </port> </service> </definitions>
```

Listing 3 - Service location

Apache CXF [25] is chosen for implementation of a web service from WSDL. After creating a WSDL file, with CXF library developing process of a JAX-WS service is divided into three steps:

1. Generate starting point code.
2. Implement the service's operations.
3. Publish the implemented service.

Generate starting point code. JAX-WS specifies a detailed mapping from a service defined in WSDL to the Java classes that will implement that service. The logical interface, defined by the `wsdl:portType` element, is mapped to a Service Endpoint Interface (SEI). Any complex types defined in the WSDL are mapped into Java classes following the mapping defined by the Java Architecture for XML Binding (JAXB) specification. The endpoint defined by the `wsdl:service` element is also generated into a Java class that is used by consumers to access endpoints implementing the service.

The **wsdl2java** command automates the generation of this code. It also provides options for generating starting point code for our implementation and an ant based makefile to build the application. The **wsdl2java** provides a number of arguments for controlling the generated code. In Table 1 are presented the classes which are commonly generated based on the information from the WSDL.

File	Description
`portTypeName.java`	*The SEI class. This file contains the java interface that service implements. This file should not be edited.*
`serviceName.java`	*The endpoint class. This file contains the Java class which clients will use to make requests on the service.*
`portTypeNameImpl.java`	*The skeleton implementation class. This class needs to be modified with concrete implementation of service*

Table 1 - WSDL generated classes

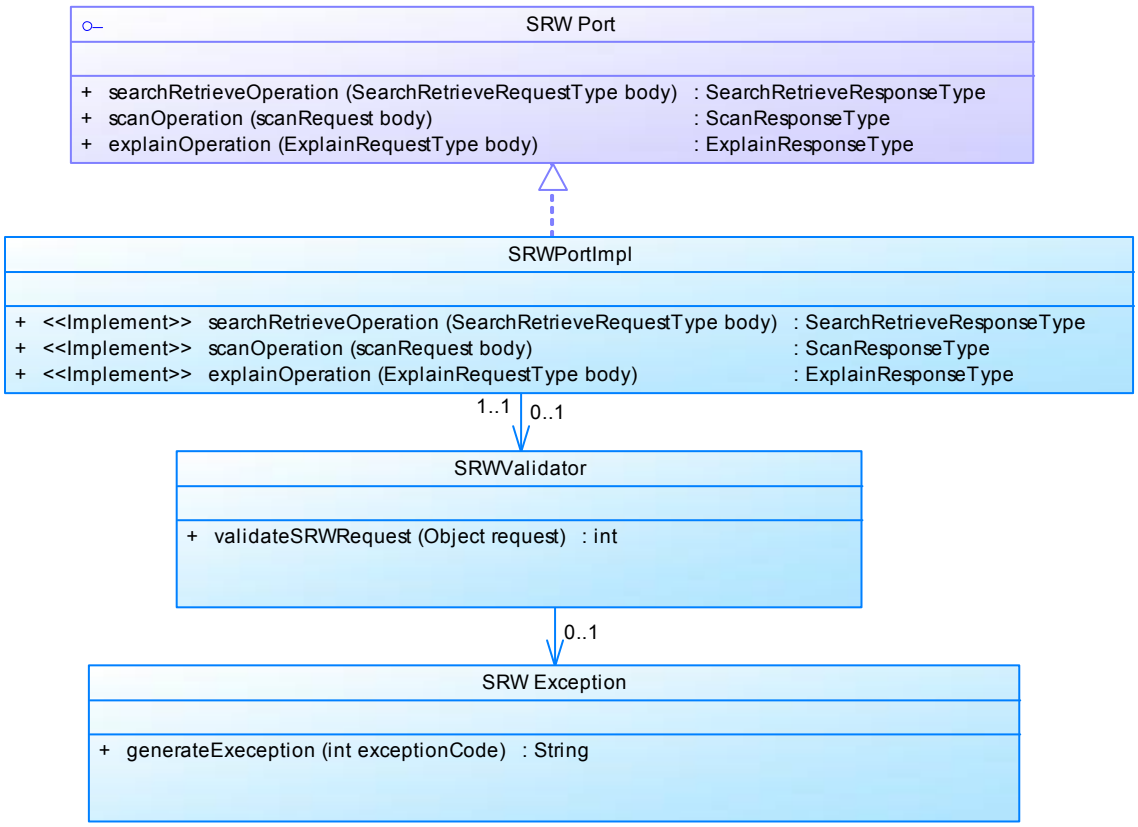

Figure 2 - SRU/W Server Side

Implementation and publishing of web services are the steps that a developer should do by changing previously generated classes.

V. SRU/W SERVICE

Class diagram of SRU/W server-side is given on the Figure 2.Generated classes are part of the SRU/W Server Side architecture. Class `SRW Port` is a Java interface which contains only declaration (prototype) of a web service functions. Implementation of service business logic is located in class `SRW Port Implementation`, where particular SRU/W service (*searchRetrive*, *scan*, *explain*) is as aimplemented separate function. Whether the requirements of clients are in accordance with the SRU/W standard, is checked by the class `SRWValidator`. Depending on the request type and disagreement with the SRU/W standard, `SRWException` class generates an appropriate message.

Communication between clients and web service in CRIS UNS system is done by exchanging SOAP messages. One of the most common scenarios is where the client sends a valid CQL query inserted as a parameter of `searchRetrieveRequest` element as shown in Listing 4.CQL query defines a request for records which a word ~~–s~~service" is contained in record titles. In SOAP request message a version of SRU/W protocol (2.0) is also specified. Parameter `<maximumRecords>` is maximum number of records which client can get in a response. Elements startRecord defines starting point in record result set (for example, if the value of the element is 3, the client wants to obtain the records, starting with the third record from the set of results). It is obviously a limit where `startRecord` must be less than or equal to the value `maximumRecords` (`startRecord <= maximumRecords`).

```
<?xml version="1.0" ?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap
/envelope/">
<searchRetrieveRequest
xmlns="http://www.loc.gov/zing/srw/"
xmlns:ns2="http://www.loc.gov/zing/cql/xcql/"
xmlns:ns3="http://www.loc.gov/zing/srw/diagnos
tic/">
<query>dc.title=&amp;quot;service&amp;quot;</query>
<version>2.0</version>
<startRecord>1</startRecord>
<maximumRecords>5</maximumRecords>
<recordPacking>xml</recordPacking>
</searchRetrieveRequest>
</soapenv:Body>
</soapenv:Envelope>
</operation>
```

Listing 4 - SRU/W SOAP request

Listing 5 outlines a response SOAP message. As it is expected, the complete response is located in separated `<searchRetrieveResponse>` element. As a part of the response, protocol version (2.0) on the server and the total number of records (`<numberOfRecords>78</ numberOfRecords>`) for a processed query are set. Main part of the response is `records` element which represents all records in accordance with the query. Also within the record there is a special element (`record`) that represents a particular record. Every single record has <recordData> where their sub-element in accordance with CRIS profile [16]. Sub-elements are related to the concrete data from records. For example, the title of the records is located in the element <dc:title> from Dublin Core context set which is a part of the CRIS profile. There is a xml schema for records within the <recordSchema> element (e.g. <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>). SRU/W request and response are both located in the SOAP Envelope section of the SOAP message.

```
<?xml version="1.0" ?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap
/envelope/">
<searchRetrieveResponse>
<version>2.0</version>
<numberOfRecords>78</numberOfRecords>
<records>
    <record>
    <recordSchema>info:srw/schema/1/dc-
v1.1</recordSchema>
    <recordData>
<dc:title>Student Service Software System,
version 2.0</dc:title>
<dc:creator>Rackovi&#263; Miloon;</dc:creator>
Rackovic Milos
<dc:creator>&Scaron;krbi&#263;
Sr&#273;an</dc:creator><br/> Škrbić Srđan
<dc:creator>Pupovac
Biljana</dc:creator><br/>Pupovac Biljana
<dc:creator>Bodroon;ki
&#381;arko</dc:creator><br/>   <
dc:publisher>, Novi Sad, Serbia</dc:publisher>
<dc:date>2007</dc:date>
<dc:type>Text</dc:type>
<dc:identifier>http://www.cris.uns.ac.rs/recor
d.jsf?recordId=6535</dc:identifier>
<dc:language>English</dc:language>
    <recordData>
    </record>
    <record>
<recordSchema>http://srw.cris.uns.ac.rs/contex
tSets/CRIS/1.0</recordSchema>
      <recordPacking>xml</recordPacking>
      <recordData>
        <srw_dc:dc>
          <dc:title>CRIS service for journals
and journal articles evaluation</dc:title>
          <dc:date>2011</dc:date>
        </srw_dc>
        <srw_cris:cris>
<cris:type>JournalArticle</cris:type>
<cris:firstAuthor>Sinisa Nikolic
</cris:firstAuthor>
<cris:abstract>This paper...</cris:abstract>
      ...
        </srw_cris:cris>
      </recordData>
    </record>
</records>
</searchRetrieveResponse>
</soapenv:Body>
</soapenv:Envelope>
</operation>
```

Listing 5 - SRU/W SOAP response

VI. CONCLUSION

In this article is presented a service for information retrieval for data from scientific research domain based on SRU/W standard.

The implementation continues the good practice of the CRIS UNS [26] by using only open source technology. SRU/W search service is modular and allows that particular components can be used and implemented in different ways, therefore it is possible to:

- Develop the external applications for search, which would be based on the SRU/W Library standard.
- Simple change of the text server, which would not affect the component SRU/W server side.
- Simultaneous search of more Text Servers, whose implementation and the physical location can be arbitrary. In this case, it is only necessary to define the appropriate mapping of CQL language into language of a Text Server.
- Make potential interoperability with various library systems since the SRU/W is de facto standard in these systems.

In the future, it is planned to enable search of scientific research data, using RESTful web service [27] standards.

ACKNOWLEDGMENT

Results presented in this paper are part of the research conducted within the Grant No. III-47003, Ministry of Science and Technological Development of the Republic of Serbia.

REFERENCES

[1] euroCRIS | Current Research Information Systems| CRIS," *euroCRIS*. [Online]. Available: http://www.eurocris.org. [Accessed: 18-Jan-2014].

[2] "Common European Research Information Format | CERIF." [Online]. Available: http://www.eurocris.org/Index.php?page=CERIFintroduction&t=1. [Accessed: 18-Jan-2014].

[3] B. Jörg, K. Jeffery, J. Dvorak, N. Houssos, A. Asserson, G. van Grootel, R. Gartner, M. Cox, H. Rasmussen, T. Vestdam, L. Strijbosch, V. Brasse, D. Zendulkova, T. Höllrigl, L. Valkovic, A. Engfer, M. Jägerhorn, M. Mahey, N. Brennan, M. A. Sicilia, I. Ruiz-Rube, D. Baker, K. Evans, A. Price, and M. Zielinski, *CERIF 1.3 Full Data Model (FDM) Introduction and Specification*. 2012.

[4] J. Dvořák and B. Jörg, "CERIF 1.5 XML - Data Exchange Format Specification," 2013, p. 16.

[5] A. N. S. I. National Information Standards Organization (U.S.), Information retrieval (Z39.50): application service definition and protocol specification : an American national standard. Bethesda, Md.: NISO Press, 2003.

[6] M. A. Hafezi, "Interoperability between library software: a solution for Iranian libraries," *Electron. Libr.*, vol. 26, no. 5, pp. 726–734, 2008.

[7] K. T. Anuradha, R. Sivakaminathan, and P. A. Kumar, "Open-source tools for enhancing full-text searching of OPACs: Use of Koha, Greenstone and Fedora," *Program Electron. Libr. Inf. Syst.*, vol. 45, no. 2, pp. 231–239, 2011.

[8] M. Zaric, D. B. Krsticev, and D. Surla, "Multitarget/multiprotocol client application for search and retrieval of bibliographic records," *Electron. Libr.*, vol. 30, no. 3, pp. 351–366, 2012.

[9] "Open Office Bibliographic project." [Online]. Available: http://www.openoffice.org/bibliographic/srw.html. [Accessed: 18-Jan-2014].

[10] "OASIS | Advancing open standards for the information society." [Online]. Available: https://www.oasis-open.org/. [Accessed: 18-Jan-2014].

[11] W. Sander-Beuermann, M. Nebel, and W. Adamczak, "Searching the CRISses," Maribor, Slovenia, 2008.

[12] K. G. Jeffery, "CRIS Architectures For Interoperation," Viena, Nov. 2007.

[13] "Current Research Information System of University of Novi Sad." [Online]. Available: http://www.cris.uns.ac.rs/. [Accessed: 18-Jan-2014].

[14] "CQL Context Set, version 1.2 - SRU Version 1.2 Specifications (SRU: Search/Retrieval via URL -- SRU, CQL and ZeeRex, Standards, Library of Congress)." [Online]. Available: http://www.loc.gov/standards/sruBob/resources/cql-context-set-v1-2.html. [Accessed: 18-Jan-2014].

[15] "Dublin Core Context Set Version 1.1 (SRU: Search/Retrieval via URL -- SRU, CQL and ZeeRex, Standards, Library of Congress)." [Online]. Available: http://www.loc.gov/standards/sru/cql/contextSets/dc-context-set.html. [Accessed: 18-Jan-2014].

[16] V. Penca, S. Nikolić, D. Ivanović, Z. Konjović, and D. Surla, "SRU/W Based CRIS Systems Search Profile," *Program Electron. Libr. Inf. Syst.*, 2014 in press.

[17] "MARC 21 Standard." [Online]. Available: www.loc.gov/marc/. [Accessed: 18-Jan-2014].

[18] "JAXB Reference Implementation — Project Kenai." [Online]. Available: https://jaxb.java.net/. [Accessed: 18-Jan-2014].

[19] "Apache Lucene." [Online]. Available: http://lucene.apache.org/. [Accessed: 18-Jan-2014].

[20] "MARCXML: The MARC 21 XML Schema." [Online]. Available: http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd. [Accessed: 18-Jan-2014].

[21] "Dublic Core Extended XML." [Online]. Available: http://dublincore.org/schemas/xmls/qdc/2006/01/06/dc.xsd. [Accessed: 18-Jan-2014].

[22] "CQL: the Contextual Query Language: Specifications (SRU: Search/Retrieval via URL, Standards, Library of Congress)." [Online]. Available: http://www.loc.gov/standards/sru/cql/. [Accessed: 18-Jan-2014].

[23] E. L. Morgan, "An Introduction to the Search/Retrieve URL Service (SRU)," *Ariadne*, no. 40, 2004.

[24] "JAX-WS Reference Implementation — Project Kenai." [Online]. Available: https://jax-ws.java.net/. [Accessed: 18-Jan-2014].

[25] "Apache CXF: An Open-Source Services Framework." [Online]. Available: http://cxf.apache.org/. [Accessed: 18-Jan-2014].

[26] D. Ivanovic, G. Milosavljevic, B. Milosavljevic, and D. Surla, "A CERIF-compatible research management system based on the MARC 21 format," *Program Electron. Libr. Inf. Syst.*, vol. 44, no. 3, pp. 229–251, 2010.

[27] L. Richardson, *RESTful web services*. Farnham: O'Reilly, 2007.