

An Analysis of a Web Service based Approach for Experimental Data Sharing

<https://doi.org/10.3991/ijoe.v14i09.8740>

Velin Kralev^(✉), Radoslava Kraleva, Nina Sinyagina
South West University, Blagoevgrad, Bulgaria
velin_kralev@swu.bg

Petia Koprinkova-Hristova, Nadejda Bocheva
Bulgarian Academy of Sciences, Bulgaria

Abstract—This paper presents the results obtained from a comparative analysis of two methods for experimental data sharing. Several works related to the topic and some approaches for processing data have been discussed. Different technologies related to the web services, ways of using them and the areas of their application are analyzed. For the purposes of the study, a web service for retrieving specific data from a behavioral experiments database was developed. The methodology and conditions for conducting the experiments are described. Two different indicators are analyzed, respectively: time to retrieve the data from a database and iteration time across all records through one loop. The results show that when retrieving thousands of records both web service based approach and an approach based on a remote database server can be used. However, when retrieving millions of records, the fastest approach was the one that uses remote database server. The obtained results show that the dynamic arrays (containing strings) iterated much faster across all data records than the dataset approach.

Keywords—web services, web methods, experimental data, database, wsdl

1 Introduction

Experimental data are an essential part of most scientific research. In order to process and analyze these data, they should be easily shared. This aspect motivated the research work presented in this paper. The purpose of this study is to analyze some data sharing approaches that can be used in different ways. The data used in this study were generated by a special hardware device (eTracker). This device can record eye movements (per person) with a sampling frequency of 1 kHz (i.e. 1000 values per 1 second). The generated values are buffered into dynamic data structures. Then, with a specialized application, this information is recorded in external files with a specific file format.

In order to provide access to the stored information (for further processing and analysis), the data must be appropriately modeled. Most often this is done when, first,

the data are processed in the way they have been stored, then they are imported into a pre-designed local or remote database. Also, these data can be accessed directly through the database management system - DBMS. This approach is used in [1], and the current work is a further development ensuing from it. Another approach through which data can be shared is based on web services. They are platform-independent applications that offer specific web methods. These web methods can be called remotely from other applications. Through combining multiple web services a new web service can be created. Thus, the web services provide the possibility of reusability and interoperability. Such an approach is presented in this paper.

Large amounts of data can be transferred through web services as well via an internal corporate network or over the Internet. In addition, this technology achieves faster and more productive communication both between users and between applications. This approach offers more effective interaction between applications that can communicate directly with each other without user support. These applications must discover, access, and interact automatically with other applications through a suitable web interface. In this way, various applications can be integrated into one large software system.

The web services are described with a specific language called WSDL (Web Service Description Language). This language provides a mechanism by which the web service definitions become available for the other applications. The communication between different web services and applications is coordinated through specific communication and transport protocols. When using web services, both applications share a common WSDL file.

Another technology related to the web services is the Simple Object Access Protocol (SOAP). It is built on the standard Hypertext Transfer Protocol (HTTP). This is why the web server can manage SOAP queries and data packets can go through firewalls. Once the interface of a web service is defined, a way for communication between it and other applications to exchange messages is needed. SOAP provides a common format for these messages. This protocol is defined at a very low abstract level and can be used by various software systems, including application servers and database management systems. SOAP specifies the form of the query and the format of its parameters. It is related only to the message and does not impose requirements to the web services themselves or their consumers [2].

2 Literature Review

More and more information systems used by different organizations provide access to data and manage different business processes using web services [3]. In order to retrieve specific information from a data repository, it is often necessary to do it in a universal way. This can be done through specific web methods that are often combined into web services. These web methods provide access to certain data in real time and can be integrated into many applications [4] and [5]. Web service based systems are used in many areas, such as e-learning platforms [6], remote laboratory access [7] and [8], and in systems for generating input data sets. [9].

The web services can be used to solve various software problems. They are used to combine different heterogeneous, decentralized and distributed systems. This means that different applications and different data formats can be dynamically integrated. An application of the web services related to the provision of a product lifecycle management process is discussed in [10]. The web services are also used to integrate different workflow management systems [11]. An approach for modeling web services using data stored in relational databases is proposed in [12]. The user interaction is based on the database logic, the defined data constraints, and the access rules. The experimental results show that some existing data-centric web applications can be used to automatically verify certain data-centric web services properties. Another issue related to the discovery of web services has been discussed in [13]. The different approaches used to solve this problem are explored and discussed. These approaches are based on keywords or unique identifiers that make it possible to find a suitable web service faster. A model is proposed to estimate the costs associated with discovering a web service, depending on the mechanisms and protocols used.

A very important aspect of the web services is its quality (Quality of Services – QoS) [14], as there are many similar web services available on the Internet. This is an important factor in the development of distributed systems, service-oriented systems, and cloud computing. A model related to the the quality of the web services based on a conditional lexicographic approach is proposed in [15]. This approach is oriented to both the web service functionality and other non-functional requirements, which, however, are important for the web service requester. In addition, specialized tools are used to monitor and control the web service quality. When the monitoring tools are not properly used and configured, this may affect the quality of service evaluation [16]. Other results obtained after analyzing the quality of the web services are presented in [17] and [18].

One of the main problems related to the web services is their security and stability against attacks over the Internet. This is especially important for web sites that provide remote procedure call (RPC). Therefore, the problem of Denial of Service (DoS) attack is discussed in [19], and various solutions are analyzed. What is more, a new model is proposed to protect against this type of attack by storing the history of the client requests. Another problem related to the security of the web services is Cross-site Scripting (XSS) attack [20]. Different security testing techniques, such as Penetration Testing and Fault Injection, are analyzed. These security techniques, combined with WS-Security (WSS), guarantee the control of access to the exchanged messages. An access control mechanism is discussed in [21]. It is implemented with a finite state machine. This approach provides a secure environment for work flow management in a particular organization.

For the web services it is very important how errors are processed and what methods are used. There is a big number of strategies for solving this problem. An overview of the well-known strategies is presented in [22]. A detailed analysis of the time to recover the web service after an error and how the different strategies affect it is made. The results show that, under different conditions, different strategies yield different results. Another study related to the analysis and processing of errors occurring during the execution of web methods is presented in [23]. The experiments show

that the proposed approach provides good results compared to other fault classification techniques.

In order the to be used by different applications, users and corporate systems, web services need to be easily discovered. The UDDI (Universal Description Discovery and Integration) technology helps organizations to find the web services and get more information about their specifics. This specific information is described in the Web Service Definition Language (WSDL). This is an XML-based language used to describe the details of the web services, such as web method names, variable types, and more. The WSDL content can be downloaded from the corresponding uniform resource identifier (URI). Some UDDI extensions that allow users to discover and call specific web methods are presented in [24]. Another web service discovery and selection method (suggested in [25]) is based on the case-based reasoning approach. The proposed method was experimentally verified, using 62 real web services. The results are good in terms of the time to retrieve the necessary information.

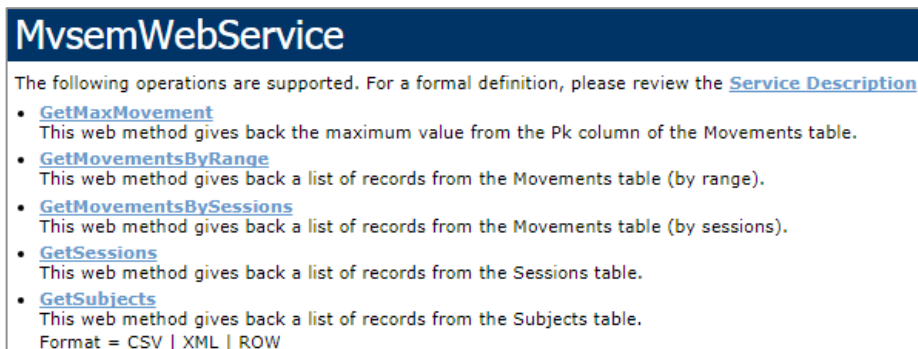
3 Development and use of the MVSEM web service

The development of the MVSEM (modelling of voluntary saccadic eye movements) web service will be briefly presented in this section. This web service will provide data from the MVSEMDM database, which is described in detail in [1]. The web service has been developed with the integrated development environment – Visual Studio (www.visualstudio.com). The database is managed by the relational database management system - MS SQL Server. Internet Information Services (www.iis.net) and ASP.NET were used to publish and use the web service.

3.1 Development and use of the MVSEM web service

The MVSEM web service provides five web methods available to users. Each of these web methods retrieves some specific information from the database and gives it back to a specific format. The data formats can be: 'CSV' - comma separated values, 'XML' - an xml element structure, and 'ROW' - an xml structure with attributes. This format is provided as an input parameter for each of the web methods. This web service is available at: <http://194.141.86.222/mvsemws/MvsemWebService.asmx> (Fig. 1).

The GetSubjects and GetSessions web methods are similar and give back corresponding lists of records from the Subjects and Sessions tables. These web methods receive an input parameter named Format (of the string type). This parameter can accept one of three values, CSV, XML or ROW, respectively. This value determines the data format that the web method will give back. The code of the GetSessions web method is shown in Fig. 2.



MvsemWebService

The following operations are supported. For a formal definition, please review the [Service Description](#).

- **GetMaxMovement**
This web method gives back the maximum value from the Pk column of the Movements table.
- **GetMovementsByRange**
This web method gives back a list of records from the Movements table (by range).
- **GetMovementsBySessions**
This web method gives back a list of records from the Movements table (by sessions).
- **GetSessions**
This web method gives back a list of records from the Sessions table.
- **GetSubjects**
This web method gives back a list of records from the Subjects table.

Format = CSV | XML | ROW

Fig. 1. The Mvsem web service in a web browser

```
01 [WebMethod (Description = "Web Method description...")]
02 public string GetSessions(string Format){
03     string q = "select Pk,FkSubject,Kind,Try from sessions";
04     SqlConnection conn = new SqlConnection(strConnString);
05     conn.Open();
06     SqlCommand cmd = new SqlCommand(q, conn);
07     SqlDataAdapter daSessions = new SqlDataAdapter(cmd);
08     DataTable dtSessions = new DataTable("sessions");
09     daSessions.Fill(dtSessions);
10     conn.Close();
11     string s = "";
12     switch (Format.ToUpper().ToString()){
13         case "CSV": {
14             s = "Pk;FkSubject;Kind;Try\n"; //header row
15             for (int i = 0; i < dtSessions.Rows.Count; i++){
16                 s = s +
17                     dtSessions.Rows[i]["Pk"].ToString() + ";" +
18                     dtSessions.Rows[i]["FkSubject"].ToString() + ";" +
19                     dtSessions.Rows[i]["Kind"].ToString() + ";" +
20                     dtSessions.Rows[i]["Try"].ToString() + "\r\n";
21             }
22             break;
23         }
24         //case "XML"; case "ROW":
25         default: {s = "No correct parameter."; break;}
26     } //switch (FormatFormat.ToUpper().ToString())
27     return s;
28 }
```

Fig. 2. Code of the GetSessions web method (in C#)

A connection object (conn) is created to connect to the database server (line 4). The connection parameters are stored in the strConnString constant, which is global for the entire web service. The connection to the database is performed by calling the Open method of the SqlConnection object (line 5). To retrieve information from the Sessions table, three objects were created, respectively SqlCommand, SqlDataAdapter and DataTable (lines 6-8). The SQL query to retrieve the information from the Sessions table is defined in line 3. This SQL statement is passed as an input parameter of the SqlCommand's constructor (line 6). Loading data from the Sessions table into the DataTable object is done by invoking the Fill method of the SqlDataAdapter object (line 9). Depending on the value of the Format input parameter, the text that will be given back to the calling application is generated (lines 12 – 26). Finally, the web method gives back the contents of the string variable s (line 27).

The GetMovementsBySessions web method gives back a list of records from the Movements table. This web method requires two input parameters. The input parameter named Format (of the string type) has the same meaning as the GetSubjects and GetSessions web methods. The other parameter is different. It is of the string type and is named Sessions. This parameter may contain a list of one or more session codes (separated by commas). The session codes can be obtained using the GetSessions web method and passed as an input parameter to this method. The code of the GetMovementsBySessions web method is shown in Fig. 3.

```
01[WebMethod (Description = "Web Method description...")]
02 public string GetMovementsBySessions(
03     string Sessions, string Format){
04     string q = "select * from movements " +
05             "where FkSession in (" + Sessions + ")";
06     //connect to server, open table and retrieving data
07     string s = "";
08     switch (Format.ToUpper().ToString()){
09         //case "CSV"; case "XML"; case "ROW"; default;
10     } //switch (Format.ToUpper().ToString())
11     return s;
12 }
```

Fig. 3. Code of the GetMovementsBySessions web method (in C#)

Connecting to the database server and retrieving the information is similar to the GetSessions web method. The difference is in constructing the query in which the value of the input parameter Sessions (line 5) is dynamically added. Depending on the Format input parameter, the text that the web method gives back as a result is generated (lines 8 – 10). Finally, the web method gives back the contents of the string variable s (line 11).

The GetMaxMovement web method gives back the maximum value from the Pk column of the Movements table. The code of this web method is shown in Fig. 4.

```
01[WebMethod (Description = "Web Method description...")]
02 public int GetMaxMovement() {
03     SqlConnection conn = new SqlConnection(strConnString);
04     conn.Open();
05     string q = "select max (Pk) from movements";
06     SqlCommand cmd = new SqlCommand(q, conn);
07     return (Int32)cmd.ExecuteScalar();
08 }
```

Fig. 4. Code of the GetMaxMovement web method (in C#)

Connecting to the database server and retrieving the information is similar to other web methods. The difference is in the given back value, which in this case is only one number (line 7). Therefore, no data are buffered into the memory.

The GetMovementsByRange web method also gives back a list of records from the Movements table, but the range of values is determined by the two input parameters - Min and Max (of the int type). The input parameter Max can also be defined using the GetMaxMovement web method. This web method will give back the maximum valid Max value. The GetMovementsByRange web method also requires an input parameter named Format (of the string type). This parameter has the same meaning as in other web methods. The code of this web method is shown in Fig. 5.

```
01[WebMethod (Description = "Web Method description...")]
02 public string GetMovementsByRange(
03     int Min, int Max, string Format) {
04     string q = "select * from movements where Pk between " +
05         Min.ToString() + " and " + Max.ToString();
06     //connect to server, open table and retrieving data
07     string s = "";
08     switch (Format.ToUpper().ToString()) {
09         //case "CSV"; case "XML"; case "ROW"; default;
10     } //switch (FormatFormat.ToUpper().ToString())
11     return s;
12 }
```

Fig. 5. Code of the GetMovementsByRange web method (in C#)

Connecting to the database server and retrieving information is similar to the GetMovementsBySessions web method. The difference consists in constructing the query in which the values of the input parameters Min and Max are dynamically added (line 5). Depending on the Format input parameter, the text is generated that the web method gives back as a result (lines 8 – 10). Finally, the web method gives back the contents of the string variable s (line 11).

3.2 Using the Mvsem web service

Three basic steps must be completed to integrate web methods into an application.

- 1) **Discovering the web service.** This is the process for finding the necessary information about the web service. This information includes the names of the web methods and their parameters. The WSDL document that is associated with the web service contains exactly this information.
- 2) **Generating an intermediate (proxy) class.** This class is an intermediary layer that encapsulates the communication process between the application and the web service. Generally, the integrated development environments offer a case tool for automatical generating of this class. They require the address of the wsdl document (or localization of a file containing the same information).
- 3) **Use the proxy class to call web methods.** The proxy class, which is a substitute for the web service, declares methods that can be called in the application. These methods in turn call the methods of the web service.

An application that uses the web methods of the Mvsem web service was developed for this study. This application has been developed with the integrated development environment - RAD Studio (www.embarcadero.com/products/rad-studio). The WSDL document that was used to generate the proxy class is posted on the Internet at:

<http://194.141.86.222/mvsemws/MvsemWebService.asmx?wsdl>

The proxy class declaration is presented in Fig. 6.

```
01 type MvsemWebServiceSoap = interface (IInvokable)
02 [ '{02000BCB-1BF5-BE02-B189-4E1D756D6FD9}' ]
03 function GetSubjects (Format: string): string; stdcall;
04 function GetSessions (Format: string): string; stdcall;
05 function GetMaxMovement: Integer; stdcall;
06 function GetMovementsBySessions (Sessions: string;
07     Format: string): string; stdcall;
08 function GetMovementsByRange (Min: Integer; Max: Integer;
09     Format: string): string; stdcall;
10 end;
11 function GetMvsemWebServiceSoap (UseWSDL: Boolean;
12     Addr: string; HTTPRIO: THTTPRIO): MvsemWebServiceSoap;
```

Fig. 6. Code of the proxy class declaration (in Delphi language)

The MvsemWebServiceSoap class implements the IInvokable interface by adding functions (methods) corresponding to the web methods of the web service (lines 3-9). Each function contains the corresponding list of parameters (as described in the WSDL document). However, the types of the parameters are specific to the particular programming language. The GetMvsemWebServiceSoap function gives back an

object of the proxy class (lines 11-12). Through this object, the application can call (in the source code) the functions that are declared in the proxy class.

A working session with the developed application is presented in Fig. 7. This application has also an additional feature to measure the execution time of each web method.

Retrieving Data from Mvsem Web Service Time: 132 241 ms 1 000 000 rows

Get Subjects as CSV As XML Get Sessions as CSV As XML Subjects & Sessions in Tree View

Pk	Initials	Sex	Age
1	ah	male	27
2	dr	male	27
3	ds	male	22
4	him	male	23
5	kap	female	32
6	kat	female	30

Pk	Sbj	Kind	Try
1	1	combine	1
2	1	flicker	1
3	1	motion	1
4	1	static	1
5	2	combined	1
6	2	combined	2

```

<Subjects>
<Subject>
  <Pk>1</Pk>
  <Initials>ah</Initials>

```

```

<Sessions>
<Session>
  <Pk>1</Pk>
  <FkSubject>1</FkSubject>

```

Subjects & Sessions in Tree View

- Subjects
 - Subject 1
 - Initials: ah
 - Sex: male
 - Age: 27
 - Session 1
 - Session 2
 - Session 3
 - Session 4
 - FkSubject: 1
 - Kind: motion
 - Try: 1

Movements for: Subject Session Range: Min= 1 Max= 1000000 Max

Format: CSV Execute Loop

Pk	FkS	Sample	EyeX	EyeY	GyroY	GyroZ	AccelX	AccelY	Screen	Mic
1	1	1	-6,998	-31,859	-1,026	-2,166	-4,349	-11,008	335	1995
2	1	2	-7,017	-31,803	-0,026	-2,166	-3,349	-9,008	335	1983
3	1	3	-7,017	-31,766	-1,026	-3,166	-2,349	-6,008	315	1978
4	1	4	-6,98	-31,766	-0,026	-2,166	-1,349	-5,008	315	2008
5	1	5	-6,943	-31,784	-0,026	-2,166	-0,349	-2,008	328	2005

Fig. 7. A working session with the web service based application

The Mvsem web service can also be integrated into applications that are developed with other development environments, such as C++ Builder or Visual Basic.

4 Experimental results

4.1 Methodology of the experiments

The aim is to analyze two different approaches for retrieving data from a database, respectively: a web service based and a database server.

For the purposes of the experiments, a second application was developed. It retrieves data directly from the database server. A working session with this application is shown in Fig. 8.

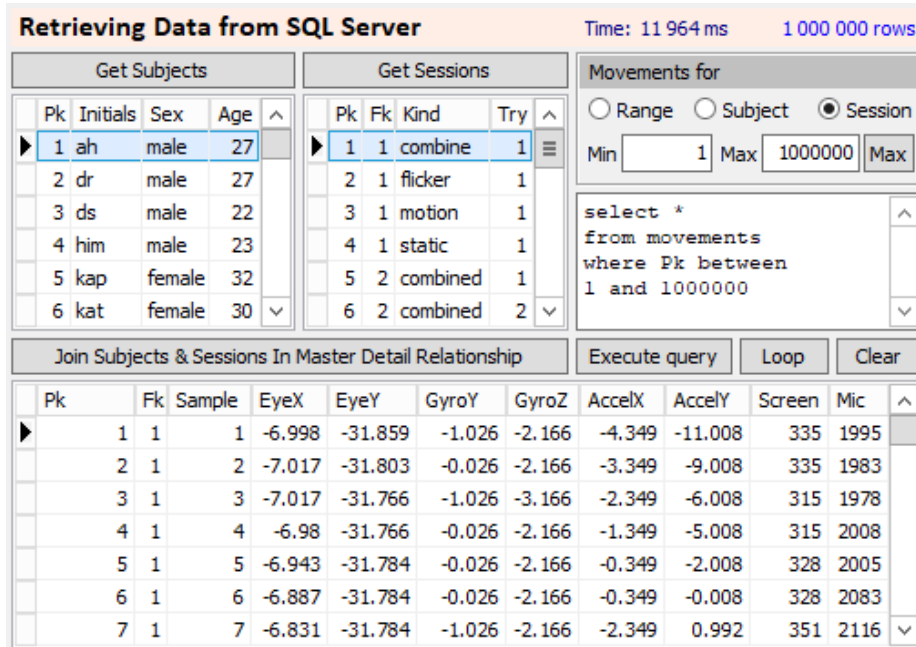


Fig. 8. A working session with the database server based application

This application has the same functionality as the web service based application. The application's user interface has appropriate controls for defining the parameters of the sql query that is sent for execution to the database server. This application also has a function to measure the execution time of the sql query.

The experimental conditions were as follows: a computer with 64-bit OS Windows 10 Pro, x64-based processor and hardware configuration: Processor: Intel (R) Core (TM) i7-6700HQ CPU at 2.60 GHz; RAM: 8GB DDR3, and remote server with 64-bit OS Windows Server 2012R2, x64-based processor and hardware configuration: Processor: Intel(R) Xeon(R) E5645 CPU at 2.40GHz 2.39GHz; RAM: 16GB DDR3. The Internet connection bandwidth is 100 Mbps (download speed: 53.57 Mbps; up-load speed: 48.71 Mbsp; ping: 9 ms).

4.2 Time to retrieve the records and time for their iteration

With the web service based application it was revealed that the time to retrieve a large number of rows (for example hundreds of thousands) is unacceptably long. It was also found out that better results are obtained when the data is retrieved in smaller blocks (e.g. 600 records in each). These data blocks are merged into a single block in this case in a dynamic array. The results of these experiments are presented in Table 1 and Fig. 9.

Table 1. Time to retrieve 15,000 records at a different data block size.

Block size	Time (ms)
200	2703
250	2437
300	2281
350	2250
400	2016
450	2172
500	2093
550	2110
600	1813
650	1968
700	2168
750	2250
800	2485

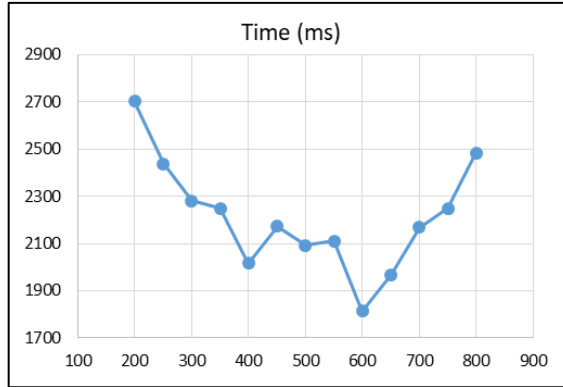


Fig. 9. Influence of the data block size on the retrieving time.

The execution time is the shortest when the data block size is set to 600.

The time to retrieve data from the Movements table was measured. Also, the time to iterate all records was measured. These values are presented in Table 2.

Table 2. Average retrieving and iterating time (in milliseconds) for different numbers of records using web service based and sql server based application.

Records	SQL Server	Web Service	Loop in DataSet	Loop in Array
100,000	1,266	13,109	1,609	141
200,000	2,203	27,203	3,125	243
300,000	4,063	40,594	4,812	375
400,000	5,734	55,765	6,422	516
500,000	6,703	61,610	8,015	609
600,000	7,766	78,875	9,750	750
700,000	8,922	87,406	11,266	875
800,000	10,159	103,969	12,766	1,016
900,000	11,453	111,703	13,984	1,156
1,000,000	12,048	130,578	15,953	1,250
2,000,000	26,078	248,265	31,328	2,531
3,000,000	40,937	434,464	47,453	4,429
4,000,000	51,391	703,831	64,047	7,175
5,000,000	63,391	1,006,479	81,625	10,260
6,000,000	76,547	1,328,552	94,625	13,543
7,000,000	85,343	1,607,548	112,688	16,387
8,000,000	97,547	1,864,756	128,953	19,828
9,000,000	111,734	2,144,469	142,203	23,397
10,000,000	129,156	2,444,695	159,531	26,906

The time required for the two applications to retrieve a certain number of records (between 100,000 and 10,000,000) was compared in Fig. 10.

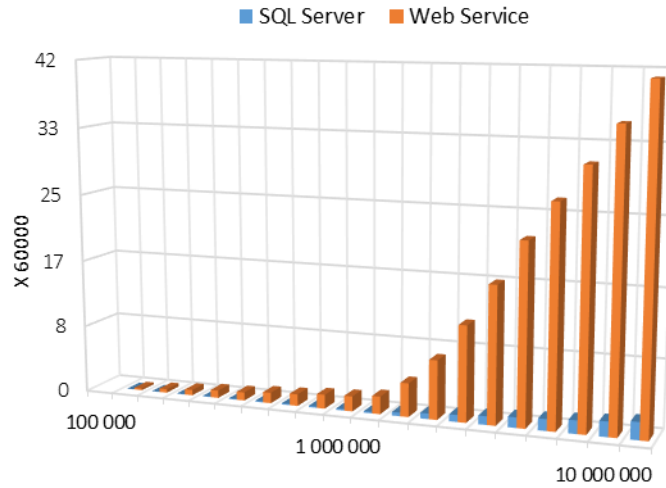


Fig. 10. Comparison of retrieving time (in minets) of a different number of records (from 100,000 to 10,000,000) for the different data retrieving approaches

The times required for the two applications to iterate all records (from 100,000 to 10,000,000) were compared in Fig. 11.

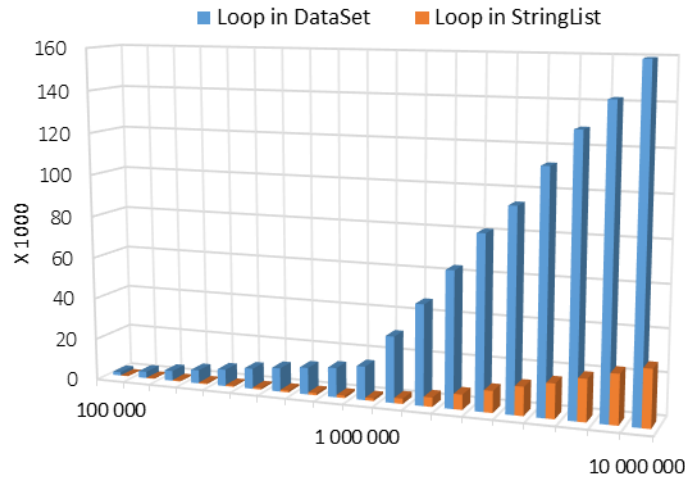


Fig. 11. Comparison of time needed to iterate over all records (in seconds) from 100,000 to 10,000,000 for both approaches

It was observed that the web service based application retrieves the records much slower than the sql server based application. This can be explained by the fact that data from the web service based application are converted several times before they reach the application. Moreover, the self-descriptive syntax of the XML language requires the use of two markers (opening and closing) to describe each value.

With a linear increase in the number of records, the retrieving time also increases linearly in all the cases. The retrieving time is acceptable for both approaches when the number of records is in the range between 100,000 and 1,000,000. However, with the web service based approach, the retrieving time significantly increases when the number of records becomes larger than 1,000,000.

From the chart in Fig. 11, it is evident that the linear increase of the records number increases linearly the time for iteration. Dynamic arrays and data sets are iterated relatively quickly. The difference in performance is seconds when the number of records is between 100,000 and 1,000,000 and minutes when the number of records is between 1,000,000 and 10,000,000. For instance, for 1,000,000 records, the iteration of a data set takes about 12.76 times more time (in seconds) than if a dynamic array is used. However, when the number of records is 10,000,000, the data set iteration takes about 5.93 times more time than the dynamic array, but in minutes.

5 Summary and conclusions

Based on the analysis of the existing data retrieving approaches, two different ways of using them, as well as the areas of their application were discussed in detail. Also, two different indicators were analyzed, respectively: time for retrieving data from a database and iteration time through all records. The obtained results showed that for retrieving a large number of records (in the order of hundreds of thousands of rows), the web service based approach and the remote database server based approach are a good choice. Regarding the data retrieving time, the approach using dynamic arrays outperforms that based on data sets.

Finally, it can be pointed out that retrieving data directly from a database server is faster than using web services. In addition, the data size can be very large and the time for retrieving is acceptable. The disadvantage is that specific data access technology is used, which may impose requirements on the operating system and computer memory. There are no such requirements when using web services. In other words, the main advantage of the web services is that they are platform independent. Depending on the specific needs of the particular study, one or the other approach or a combination of them may be chosen.

6 Acknowledgment

The reported work is a part of and was supported by project № DN02/3 "Modelling of voluntary saccadic eye movements during decision making" funded by the Bulgarian Science Fund.

7 References

- [1] Krалева, R., Krалев, V., Sinyagina, N., Koprinkova-Hristova, P., Bocheva, N. (2018). Design and analysis of a relational database for behavioral experiments data processing. *International Journal of Online Engineering*, 14(2): 117-132. <https://doi.org/10.3991/ijoe.v14i02.7988>
- [2] Gurugé, A. (2004). *Web services: theory and practice*. Elsevier.
- [3] Turc, T. (2015). Using WEB Services in SCADA Applications. *Procedia Technology*, 19: 584-590. <https://doi.org/10.1016/j.protcy.2015.02.083>
- [4] Grahl, M., Bluhm, T., Grun, M., Hennig, C., Holtz, A., Krom, J. G., Kuhner, G., Laqua, H., Lewerentz, M., Riemann, H., Spring, A., Werner, A. (2017). Archive WEB API: A web service for the experiment data archive of Wendelstein 7-X. *Fusion Engineering and Design*, 123: 1015-1019. <https://doi.org/10.1016/j.fusengdes.2017.02.047>
- [5] Niknam, M., Karshenas, S. (2015). Sustainable Design of Buildings using Semantic BIM and Semantic Web Services. *Procedia Engineering*, 118: 909-917. <https://doi.org/10.1016/j.proeng.2015.08.530>
- [6] Akrouf, S., Merabet, A., Maza, A., Boubetra, D., Selmani, L., Boubetra, A., Mouhoub, N. (2014). Web services for virtual simulation. *International Journal of Online Engineering*, 10(5): 9-11. <https://doi.org/10.3991/ijoe.v10i5.3796>
- [7] Tawfik, M., Salzmann, C., Gillet, D., Lowe, D., Saliyah-Hassane, H., Sancristobal, E., Castro, M. (2014). Laboratory as a service (LaaS): A novel paradigm for developing and implementing modular remote laboratories. *International Journal of Online Engineering*, 10(4): 13-21. <https://doi.org/10.3991/ijoe.v10i4.3654>
- [8] Wang, N., Chen, X., Song, G., Parsaei, H. (2015). An Experiment Scheduler and Federated Authentication Solution for Remote Laboratory Access. *International Journal of Online Engineering*, 11(3): 20-26. <http://dx.doi.org/10.3991/ijoe.v11i3.4554>
- [9] Krалева, V., Krалева, R. (2011). Web Service Based System for Generating Input Data Sets. *Proceedings of the Fourth International Scientific Conference*, 2: 49-56.
- [10] Franke, M., Klein, K., Hribernik, K., Lappe, D., Veigt, M., Thoben, K-D. (2014). Semantic Web Service Wrappers as a Foundation for Interoperability in Closed-loop Product Lifecycle Management. *Procedia CIRP*, 22: 225-230. <https://doi.org/10.1016/j.procir.2014.07.020>
- [11] Mahmoud, T., Rapp, B., Vliet, S. (2013). Web Service Integration within Next Generation CEMIS. *Procedia Technology*, 9: 282-290. <https://doi.org/10.1016/j.protcy.2013.12.032>
- [12] Fang, Z., Liao, L., Chen, R. (2013). Automatic Verification of Data-centric Web Service Specifications. *IERI Procedia*, 4: 93-98. <https://doi.org/10.1016/j.ieri.2013.11.015>
- [13] Ayad, S., Kazar, O., Benharkat, N. (2014). Evaluating the Energy Consumption of Web Services Protocols in Ad Hoc Networks. *AASRI Procedia*, 7: 8-13. <https://doi.org/10.1016/j.aasri.2014.05.021>
- [14] Rajeswari, M., Sambasivam, G., Balaji, N., Saleem Basha, M. S., Vengattaraman, T., Dhavachelvan, P. (2014). Appraisal and analysis on various web service composition approaches based on QoS factors. *Journal of King Saud University - Computer and Information Sciences*, 26(1): 143-152. <https://doi.org/10.1016/j.jksuci.2013.08.003>
- [15] Iordache, R., Moldoveanu, F. (2014). QoS-Aware Web Service Semantic Selection Based on Preferences. *Procedia Engineering*, 69: 1152-1161. <https://doi.org/10.1016/j.proeng.2014.03.104>
- [16] Ruiz, J. Z., Rubira, C. M. (2016). Quality of Service Conflict During Web Service Monitoring: A Case Study. *Electronic Notes in Theoretical Computer Science*, 321: 113-127. <https://doi.org/10.1016/j.entcs.2016.02.007>

- [17] Amr S. Abdelfattah, Tamer Abdelkader, El-Sayed M. El-Horbaty. (2017). RSAM: An enhanced architecture for achieving web services reliability in mobile cloud computing. *Journal of King Saud University - Computer and Information Sciences*, available online 10 March 2017. <https://doi.org/10.1016/j.jksuci.2017.03.002>
- [18] Aljazzaf, Z. (2015). Bootstrapping quality of Web Services. *Journal of King Saud University - Computer and Information Sciences*, 27(3): 323-333. <https://doi.org/10.1016/j.jksuci.2014.12.003>
- [19] Venkatesan, S., Saleem Basha, M. S., Chellappan, C., Vaish, A., Dhavachelvan, P. (2013). Analysis of accounting models for the detection of duplicate requests in web services. *Journal of King Saud University - Computer and Information Sciences*, 25(1): 7-24. <https://doi.org/10.1016/j.jksuci.2012.05.003>
- [20] Salas, M. I. P., Martins, E. (2014). Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security. *Electronic Notes in Theoretical Computer Science*, 302: 133-154. <https://doi.org/10.1016/j.entcs.2014.01.024>
- [21] Thirumaran, M., Dhavachelvan, P., Aishwarya, D., Shanmugapriya, R. (2013). Finite State Machine based Access Control Mechanism for Web Service Work Flow Management. *IERI Procedia*, 4: 391-397. <https://doi.org/10.1016/j.ieri.2013.11.056>
- [22] Angarita, R., Cardinale, Y., Rukoz, M. (2014). Reliable Composite Web Services Execution: Towards a Dynamic Recovery Decision. *Electronic Notes in Theoretical Computer Science*, 302: 5-28. <https://doi.org/10.1016/j.entcs.2014.01.018>
- [23] Han, X., Li, B., Wong, K-F., Shi, Z. (2016). Exploiting structural similarity of log files in fault diagnosis for Web service composition. *CAAI Transactions on Intelligence Technology*, 1(1): 61-71. <https://doi.org/10.1016/j.trit.2016.03.006>
- [24] Marwaha, P., Banati, H., Bedi, P. (2013). UDDI Extensions for Temporally Customized Web Services. *Procedia Technology*, 10: 184-190. <https://doi.org/10.1016/j.protcy.2013.12.351>
- [25] Renzis, A. D., Garriga, M., Flores, A., Cechich, A., Zunino, A. (2016). Case-based Reasoning for Web Service Discovery and Selection. *Electronic Notes in Theoretical Computer Science*, 321: 89-112. <https://doi.org/10.1016/j.entcs.2016.02.006>

8 Authors

Velin Kralev is an assistant professor of Computer Science at the Faculty of Mathematics and Natural Sciences, South-West University "Neofit Rilski", Blagoevgrad, Bulgaria. He defended his PhD Thesis in 2010. His research interests include database systems development, optimization problems of the scheduling theory, graph theory and component-oriented software engineering. He is an editorial board member of the *International Journal of Advanced Computer Research (IJACR)*.

Radoslava Kraleva is an assistant professor of Computer Science at the Faculty of Mathematics and Natural Sciences, South-West University "Neofit Rilski", Blagoevgrad, Bulgaria. She defended her PhD Thesis "Acoustic-Phonetic Modeling for Children's Speech Recognition in Bulgarian" in 2014. Her research interests include child-computer interaction, speech recognition, mobile app development and computer graphic. She is an editorial board member of the "International Journal of Advanced Computer Research" and of the journal "Perspectives of Innovations, Economics and Business". She is a reviewer of *iJET*, "International Journal on Advanced Science, Engineering and Information Technology", "Computer Standards & Inter-

faces”, “Journal of King Saud University - Computer and Information Sciences”, and many others.

Nina Sinyagina has received her PhD degree in St. Petersburg Technical University in 1972. From 1998 she was Director of a research group in the Institute for parallel processing of information, Bulgarian Academy of Sciences. Since 2004 until now she has been a professor at the South-West University, Blagoevgrad. Her main research areas are: Pattern recognition, Artificial Intelligence, Computer architectures, Database, Computer Security, Operational Systems, and Computer Networks.

Petia Koprinkova-Hristova received MSc degree in Biotechnics from the Technical University - Sofia in 1989 and PhD degree on Process Automation from Bulgarian Academy of Sciences in 2001. Since 2003 she has been an Associate Professor at the Institute of Control and System Research and from January 2012 - at the Institute of Information and Communication Technologies, Bulgarian Academy of Sciences. Her main research interests are in the field of Intelligent Systems using mainly neural networks, fuzzy and neuro-fuzzy approaches. Currently she is a member of the European Neural Network Society (ENNS) executive committee and member of the Union of Automatics and Informatics in Bulgaria.

Nadejda Bocheva is an Associate Professor at the Institute of Neurobiology (former Institute of Physiology), Bulgarian Academy of Sciences. She received a PhD degree in biology in 1986 from the Institute of Physiology, Bulgarian Academy of Sciences. In 2006 she became an Associate Professor in psychophysiology. At present she is head of Department of Sensory Neurobiology at the Institute of Neurobiology. Her research interests are in human visual information processing, spatial vision, motion perception, visual recovery of 3D shape, cognitive neuroscience and aging. She is a Fulbright fellow and was awarded a Fogarty international collaborative Award in 2002. She is a member of the American Psychological Association and of the Sofia section of the Bulgarian Physiological Society.

Article submitted 15 April 2018. Final acceptance 18 May 2018. Final version published as submitted by the authors.