Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 21 No. 5 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v21i05.53819

PAPER

A Web Service-Based Application for Processing EEG Experimental Data Generated in Response to Visual Stimuli

Velin Kralev[1](✉), Radoslava Kraleva[1], Petia Koprinkova-Hristova[2], Nadejda Bocheva[2], Miroslava Stefanova[2]

[1]South-West University, Blagoevgrad, Bulgaria

[2]Bulgarian Academy of Sciences, Sofia, Bulgaria

velin_kralev@swu.bg

ABSTRACT

This paper presents the results of a comparative analysis of different approaches to retrieving experimental data. The different methods of processing the data and their applications are analyzed. For the aims of the study, a database was designed to store, analyze, and process experimental data generated in response to visual stimuli. The methodology and conditions for conducting the experiments are detailed. The experiments utilized real data, which can be extracted by calling the corresponding web methods. For the experiments, 12 independent tests were conducted, and six operations related to extracting, processing, and analyzing the experimental data were examined. The results indicate that data extraction is fastest when using a direct approach, specifically by executing a query directly on the database server. Based on the results, the two slowest operations were found to be related to data extraction using web methods. The first variant describes the data in text format, while the second uses XML. The second approach is significantly slower in data retrieval, by about 30%. In addition to data retrieval operations, another data processing operation was also analyzed once the data was buffered in the address space of the client application. It was found that these operations are performed in an acceptable time—slower than direct data retrieval from the database server but faster than data retrieval using web methods.

KEYWORDS

experimental data, database, web services, web methods, data processing, data analysis, human eye movements, software development

1 INTRODUCTION

The different types and sizes of experimental data, generated in various scientific studies, necessitate the search for new, more flexible, and faster methods for storing, sharing, and processing these data. The goal is to provide a way for easy but secure

Kralev, V., Kraleva, R., Koprinkova-Hristova, P., Bocheva, N., Stefanova, M. (2025). A Web Service-Based Application for Processing EEG Experimental Data Generated in Response to Visual Stimuli. *International Journal of Online and Biomedical Engineering (iJOE)*, 21(5), pp. 142–159. https://doi.org/10.3991/ijoe.v21i05.53819

Article submitted 2024-12-12. Revision uploaded 2025-01-21. Final acceptance 2025-01-21.

access to the collected information from the conducted experiments. Collected data needs to be pre-processed and converted to an appropriate format and then stored in a database (or data warehouses). This data pre-processing process may also include pre-importing the data into a dedicated local or remote data store [1]. Typically, this approach uses a relational database management system (RDBMS).

Multiple approaches that are oriented towards the use of data management systems are discussed in various studies. They differ in their purpose and operation; for example, a secure approach for authenticating access to experimental data is presented in [2]. Other studies focus on the process of designing and organizing data sets, for example, methods for evaluating data efficiency and integrity [3] or methods for detecting inconsistencies between data and the files in which they are stored [4]. A method for designing a distributed relational database scheme that is oriented toward using queries and replications to update data in the database is presented in [5]. Various approaches to error detection in data warehouse modeling have been used successfully in [6].

The entity-relationship (ER) approach is also used in data modeling and database design [7]. This approach makes it possible to transform the ER schema into a relational database schema or into a class definition. In this way, it is possible to generate Structured Query Language (SQL) code to automatically create the database and the metadata associated with its description, including relationships and constraints. There are other approaches to designing databases for a different application [8]. The modeling, respectively the design of the data affects the efficiency of the operations of filtering and searching for information in data stores [9]. Security of information and data processing is another aspect that is related to databases and that is discussed in the scientific literature [10]. However, increasing the security of the data, for example, by encryption, reduces the performance of the DBMS while also increasing the time to retrieve and store data in the database [11].

In addition to centralized data storage servers, data can be shared and accessed through web services technology. Web services are applications that are independent and provide various web functions (methods). These functions can be called remotely from functions of other web services or from functions of other applications. One possible approach to accessing large amounts of data is to use database servers that can be accessed via internal networks or the Internet. Another approach is to store the data centrally but make it accessible only through web service functions. Another important aspect of this approach is the ability to communicate between the applications themselves. This is possible because two different applications can call shared web functions to communicate with each other. This approach is more efficient because it allows applications to communicate with each other without the need for a user. By combining several relatively independent applications that share common web services and functions, it is possible to build a larger integrated system [12].

The Web Services Description Language (WSDL) provides the ability to describe any web service in such a way that it is accessible to other applications or other web services. When two applications or two web services or an application and a web service use common web functions, they must use a common WSDL description. When the functions of a web service are defined, a protocol is needed for communication between the applications and the web services. The Simple Object Access Protocol (SOAP) was developed for this purpose. Because this protocol is based on the Hypertext Transfer Protocol (HTTP), the web servers can use SOAP requests over the Internet. In fact, SOAP requests are messages with a standardized format (and list of parameters). Through these messages, web services and applications communicate.

This requires web services to have a certain quality (Quality of Services-QoS) [13], as they are used over the Internet in distributed systems, in cloud computing systems, and in integrated systems using mainly web services [14]. Different methods for analyzing the quality of web services are used. In this way, web services can be further configured to increase their quality [15].

Another important aspect of web services is their security against network attacks. Various defense mechanisms have been developed and presented, for example, against Denial of Service (DoS) attacks [16] and Cross-site Scripting (XSS) attacks [17]. Another access control mechanism for message exchange is presented in [18]. These and other similar mechanisms, combined with WS-Security (WSS), ensure security when working with web services. When errors occur while working with web services, various methods are used to fix them. Commonly used strategies in such situations are discussed in [19]. Another approach for real-time error analysis is presented in [20]. The results show that the proposed approaches ensure a stable state of web services after recovery from errors.

Integrated systems using web functions are applicable in various fields, for example, in e-learning systems and research laboratories [21], for project management, or for the integration of different business processes [22]. They are also used to unify inhomogeneous data in distributed systems. Using this approach, it is possible for one application to use the data in one format and another application to use the data in another format. Also, data can be shared through conversion, through web functions, or directly retrieved from the database [23]. Such approaches will also be used in the present study, which is oriented to the processing and analysis of data generated in experiments with visual stimuli.

The data to be processed and stored is obtained by conducting experimental sessions with people. Visual stimuli are presented to each subject on a monitor. The subject's task is to recognize the correct direction of a stimulus that presents an outline of a figure that is displaced from the center to the left or right in option one, or up or down in option two. The subject indicates the recognized direction of the stimulus, which is written to a result file (with the extension .dat). Then, the subject sees the next stimulus, and so on. The number of visualized stimuli was predetermined, but their order was randomly generated. The main task of the experiment is to collect data on human visual perception of moving objects. During the experiment, the values obtained from the electroencephalogram (EEG) of each person are recorded. The raw data is stored in a file that contains the following information:

1. Stimulus duration (in milliseconds).
2. Change the marker on the correct or wrong answer by the subject. This marker is entered in the result file when a correct answer is followed by a wrong one, and vice versa when a wrong answer is followed by a correct one.
3. The change in the time of presentation of the stimulus, i.e., whether it is increasing or decreasing. When the stimulus presentation time increases (relative to previous stimuli), a (+) sign is entered in the result file. Accordingly, when the stimulus presentation time decreases, a (–) sign is entered in the result file.
4. The type of stimulus. Depending on the direction of the stimulus, there are four types, labeled as follows: L-left, R-right, U-up, and D-down.
5. The recognized type of stimulus, depending on the subject's response, corresponds to L-left, R-right, U-up, and D-down.
6. The time required for the subject's response to recognize the direction of the stimulus.

Each stimulus presents the task of recognizing the direction of movement of a set of dots in a certain direction. The stimuli were pre-generated but presented to each subject in a different order [24]. These types of visual tasks are actively researched and related to analyzing the decision-making process of subjects [25]. Approaches are used to model various processes [26], which are oriented towards the application of artificial intelligence methods [27]. Other methods are based on neural networks [28]. A subclass of these tasks that involves visual stimuli and where it is necessary to recognize one of two possible responses (in this case two possible directions of motion) is presented in [29]. The main aspect of these studies is to determine the influence of various factors on the subject's decision-making process [30]. Other analysis methods include investigating the influence of decision time on its accuracy [31], as well as multi-criteria methods presented in detail in [32]. During some experimental sessions, EEG information is also stored and analyzed.

Extracting information from these experiments requires that the generated data be processed and stored in an appropriate format so that it can be easily (but securely) accessed and analyzed. Different approaches and technologies can be used to organize and access the data, for example: relational databases, object-oriented databases, object-relational databases, graph-based databases, and/or web services [33, 34, 35]. A relational database was created to store the data from the conducted experiments. In order for a user to retrieve information from the database, it is necessary to have at least read privilege on the database tables, i.e., the user must have at least the dbreader role. The approach of maintaining multiple users (logins) and setting their permissions is not convenient. Since the data is read-only, users can use a generic web service that has permission to read the data and that provides web methods (functions) to retrieve it. This approach is the subject of analysis in the present study. Web functions can be used efficiently to retrieve data to be sent to the calling modules [36]. This enables the data to be converted also in different formats [37]. Since this approach provides opportunities to create scalable systems [38] with multiple functions, it will also be the subject of analysis in the present study.

2 MATERIALS AND METHODS

2.1 Design of the PPSCPCRVIDB database

The structure of the created database named "Post-Perceptual Stages of Cognitive Processing and Conscious Representations of Visual Information" (PPSCPCRVIDB) is shown in Figure 1.

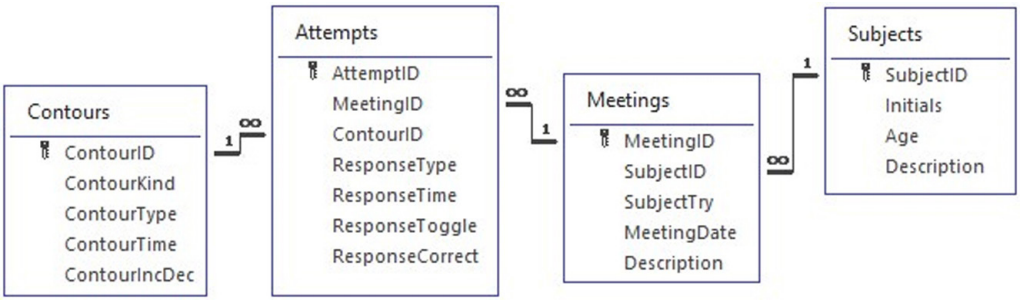

Fig. 1. Schema of the database "**PPSCPCRVIDB**"

Each shape on the scheme contains one relation of the database. The four relations created are:

1. The "Subjects" table (relation) stores data about the subjects in the experiments. The attributes of these subjects are their identification number (the attribute "SubjectID"), their initials (the attribute "Initials"), their age (the attribute "Age"), and some additional information, if it is necessary to store one (attribute "Description"). However, this information is of type multi-line text field and therefore cannot be used when sorting, filtering, and searching for information in the "Subjects" table. The data in the "Description" attribute can only be used as additional information when analyzing the experimental results.
2. The "Contours" table stores data about the stimulus implementation, for example: the kind of the stimulus (the "ContourKind" attribute with possible values "Horizontal" or "Vertical"); the type of stimulus (the ContourType attribute with possible values respectively: in horizontal kind–"Left" or "Right" in vertical kind–"Up" or "Down"); the time of stimulus (the "ContourTime" attribute in milliseconds); and the change in the time of presentation of the stimulus–whether it increases or decreases (the "ContourIncDec" attribute).
3. This "Meetings" table stores data about the sessions. For this aim, the MeetingDate attribute stores the date of the experiment session. The attribute "SubjectID" stores the subject identification number. This attribute implements a one-to-many relationship between the "Subjects" and "Meetings" tables.
4. The "Attempts" table stores data about the attempt conducted by the subject. The subject's response to a given stimulus is stored in the attribute "ResponseType." This is done by a chain of relationships between keys (primary and foreign), for example: SubjectID - SubjectID (between "Subjects" and "Meetings" relationis); MeetingID - MeetingID (between "Meetings" and "Attempts" relations); ContourID - ContourID (between "Attempts" and "Contours" relations).

The most significant relation in the presented database is the "Attempts." A many-to-many relationship between the "Meetings" and "Contours" relations is carried out through the "MeetingID" and "ContourID" attributes because, first, many attempts can be conducted by one subject, and second, every contour (stimulus) can be shown to many subjects. This type of join is implemented with another table that has foreign keys that link to the primary keys of the other two tables. In this case these are the MeetingID and ContourID attributes from the "Attempts" table, which is associative. The code for creating the "Attempts" relation is shown in Figure 2.

```
create table [Attempts](
  [AttemptID] [int] not null primary key,
  [MeetingID] [int] not null foreign key
     references [Meetings]([MeetingID]),
  [ContourID] [int] not null foreign key
     references [Contorus]([ContourID]),
  [ResponseType] [varchar(10)] not null,
  [ResponseTime] [datetime] not null,
  [ResponseToggle] [varchar(10)],
  [ResponseCorrect] [bit] not null);
```

Fig. 2. The SQL code for creating the table "Attempts"

The tables thus designed and constructed fulfill the requirements of the relational data model in terms of first, second, and third normal forms. This is because all tables have a defined primary key, contain no multivalued attributes, and all attributes are atomic (indivisible). When these requirements are met, according to the relational data model, the corresponding table (relation) is in first normal form (1NF). Also, designed tables fulfill the requirements of second normal form (2NF) because there are no partial functional dependencies between the non-key attributes and the key attribute. Furthermore, the designed tables are also in third normal form (3NF), since they do not contain transitive functional dependencies between their attributes.

2.2 Implementation of the PPSCPCRVI web service

The implementation of the PPSCPCRVI (Post-Perceptual Stages of Cognitive Processing and Conscious Representations of Visual Information) web service will be presented in this section. This service will provide information from the PPSCPCRVI database, which is described in the previous section. The service has been implemented with the Visual Studio development environment. The database is created by the management system—MS SQL Server. Internet Information Services and ASP.NET were used to publish the PPSCPCRVI web service.

The PPSCPCRVI web service provides two methods. Each one of these methods retrieves specific data from the database and returns it to a concrete format. The specific formats can be: 'TEXT'—a format based on XML tags with attributes and 'XMLF'—a format based on XML tags with elements. Each of the two types of formats is set as an input parameter to each of the web methods. The PPSCPCRVI service is available on the Internet and can be tested at: http://194.141.86.222/ppscpcrvi or PpscpcrviWebService.asmx (see Figure 3).

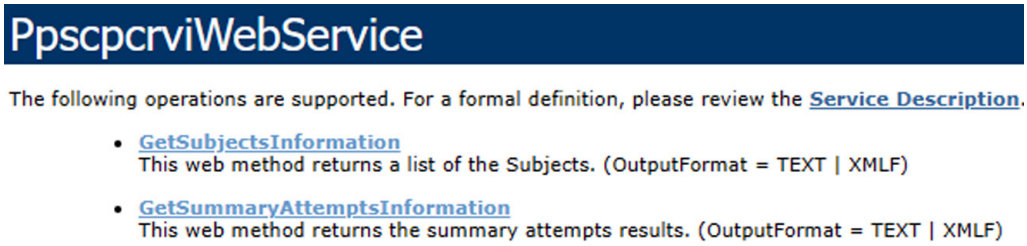

Fig. 3. The main page of the web service PPSCPCRVI

The GetSubjectsInformation function returns a corresponding lists of rows from the Subjects table. This function receives a variable as input value with name output format. This variable can contain one of two values, TEXT or XMLF, respectively. This parameter specifies the format of the information that the function will return. The short version of the code of the GetSubjectInformation function is presented in Figure 4.

```
01 public string GetSubjectsInformation(string OutputFormat){
02  string Sql = "SELECT SubjectID, Initials, Age, Description
03                FROM Subjects";
04  SqlConnection SqlCn = new SqlConnection(ConnectionString);
05  SqlCn.Open();
06  SqlCommand SqlCmd = new SqlCommand(Sql, SqlCn);
07  SqlDataAdapter DASubjects = new SqlDataAdapter(SqlCmd);
08  DataTable DTSubjects = new DataTable("Subjects");
09  DASubjects.Fill(DTSubjects);
10  SqlCn.Close();
11 string RText = "";
12 switch (OutputFormat.ToUpper().ToString()){
13  case "TEXT":{RText = "<Subjects>";
14   for (int i = 0; i < DTSubjects.Rows.Count; i++){
15    RText = RText + "<Row " +
16     "SubjectID=\"" + DTSubjects.Rows[i]["SubjectID"] +
17     "Initials=\"" + DTSubjects.Rows[i]["Initials"] +
18     "Age=\"" + DTSubjects.Rows[i]["Age"] + "\" " +
19     "Description=\"" + DTSubjects.Rows[i]["Description"] +
20     "/>";} RText = RText + "</Subjects>"; break;}
21  case "XMLF":{RText = "<Subjects>";
22   for (int i = 0; i < DTSubjects.Rows.Count; i++){
23    RText = RText + "<Subject>" +
24    "<SubjectID>" + DTSubjects.Rows[i]["SubjectID"] +
25    "<Initials>" + DTSubjects.Rows[i]["Initials"] +
26    "<Age>" + DTSubjects.Rows[i]["Age"] +
27    "<Description>" + DTSubjects.Rows[i]["Description"] +
28    "</Subject>";} RText = RText + "</Subjects>"; break;}
29  default: {RText = "No correct parameters."; break;}}
30  return RText;}
```

Fig. 4. The code of the GetSubjectInformation function

An instance (SqlCn) is defined to connect to the database server (line 04). The connection information is stored in the Connection String parameter, which is global for the service. To access the data in the database, ppscpcrvidb can call the Open function of the SqlCn instance (line 05). To retrieve data from the Subjects relation, there were instances defined: SqlCmd (of type SqlCommand), DA Subjects (of type SqlData Adapter), and DT Subjects (of type Data Table (lines 06–08)). The Sql query (in structured query language - SQL) to get the data from the Subjects table is generated on lines 02 and 03. This query is passed as a parameter of the Sql Cmd's method (line 06). Getting information from the Subjects relation in the DT Subjects instance is done by calling the Fill method of the SD Subjects instance (line 09). Depending on the value of the Output Format variable, the returned result is generated in a specific format (lines 12–28). Finally, the function returns the value of the R Text parameter.

The GetSummaryAttemptsInformation function is similar to the Get Subjects Information function and returns a lists of rows that are the result of executing a specific sql query. This query retrieves information from the PPSCPCRVIDB database, which is stored in four tables: Attempts, Contours, Meetings, and Subjects. These tables are joined using the SQL JOIN command using the corresponding primary and foreign key values of each of these tables. For example, the Attempts and Contours tables are joined by the ContourID field, as this field in the Contours table is a primary key and in the attempts table it is a foreign key. Similarly, the Attempts and Meetings tables are joined by the MeetingID field, as this field in the Meetings table is a primary key and in the attempts table it is a foreign key. Similarly, the

Meetings and Subjects tables are joined by the SubjectID field, as this field in the Subjects table is a primary key and in the Meetings table it is a foreign key. The Get Summary Attempts Information function also receives one input parameter named Output Format, which has the same meaning and usage as the one in the Get Subjects Information function. The two meanings of the Output Format parameter, TEXT and XMLF respectively, are used in an analogous way. A shortened version of the Get Summary Attempts Information function code is presented in Figure 5.

```
01 public string
02 GetSummaryAttemptsInformation(string OutputFormat){
03 string Sql =
04 "SELECT Attempts.AttemptID,Subjects.SubjectInitials,
05         Contours.ContourKind, Meetings.SubjectTry,
06         Contours.ContourID, Contours.ContourTime,
07         Attempts.ResponseToggle, Contours.ContourIncDec,
08         Contours.ContourType, Attempts.ResponseType,
09         Attempts.ResponseTime, Attempts.ResponseCorrect
10  FROM Attempts JOIN Contours ON
11    Attempts.ContourID = Contours.ContourID JOIN Meetings ON
12    Attempts.MeetingID = Meetings.MeetingID JOIN Subjects ON
13    Meetings.SubjectID = Subjects.SubjectID
14 ORDER BY Attempts.AttemptID ASC;
15  SqlConnection SqlCn = new SqlConnection(ConnectionString);
16  SqlCn.Open();
17  SqlCommand SqlCmd = new SqlCommand(Sql, SqlCn);
18  SqlDataAdapter DA = new SqlDataAdapter(SqlCmd);
19  DataTable DT = new DataTable("ResultsView"); DA.Fill(DT);
20  SqlCn.Close();
21 string RText = "";
22 switch (OutputFormat.ToUpper().ToString()){
23  case "TEXT":{RText = "<ResultsView>";
24   for (int i = 0; i < DT.Rows.Count; i++){
25  RText = RText + "<Row " +
26 "AttemptID=\"" + DT.Rows[i][0].ToString() + "\" " +
27 "SubjectInitials=\"" + DT.Rows[i][1].ToString() + "\" " +
28 "ContourKind=\"" + DT.Rows[i][2].ToString() + "\" " +
29 "SubjectTry=\"" + DT.Rows[i][3].ToString() + "\" " +
30 "ContourID=\"" + DT.Rows[i][4].ToString() + "\" " +
31 "ContourTime=\"" + DT.Rows[i][5].ToString() + "\" " +
32 "ResponseToggle=\"" + DT.Rows[i][6].ToString() + "\" " +
33 "ContourIncDec=\"" + DT.Rows[i][7].ToString() + "\" " +
34 "ContourType=\"" + DT.Rows[i][8].ToString() + "\" " +
35 "ResponseType=\"" + DT.Rows[i][9].ToString() + "\" " +
36 "ResponseTime=\"" + DT.Rows[i][10].ToString() + "\" " +
37 "ResponseCorrect=\"" + DT.Rows[i][11].ToString() + "\" " +
38 "/>";} RText = RText + "</ResultsView>"; break;}
39  case "XMLF":{RText = "<ResultsView>";
40   for (int i = 0; i < DT.Rows.Count; i++){
41     /* Add items in XML format */ }
42     RText = RText + "</ResultsView>"; break;}
43  default: {RText = "No correct parameters."; break;}}
44  return RText;}
```

Fig. 5. The code of the GetSummaryAttemptsInformation function

Another SqlConnection object (SqlCn) is declared to connect to the database server (line 15). The connection information is obtained from the ConnectionString global constant. Analogous to the GetSubjectsInformation function and the GetSummary AttemptsInformation function, the data is retrieved from the database and buffered in memory by calling the Fill method (line 19) of the DA object (of the SqlDataAdapter class), with which it is loaded into the DT object (of the DataTable class). Depending on the value of the OutputFormat parameter, the returned result is generated in a specific format (lines 23–42). Finally, the function returns the generated information to the RText parameter, with the information structured according to the specified formatting.

2.3 Using the PPSCPCRVI web service

Once a web service is created and its methods are published to be accessible, several preliminary phases need to be completed before that web service can be used by external applications. Before the functions of a web service can be called from an application, the following steps need to be performed.

The first step is to find the necessary information about the web services on the internet. This step involves the process of discovering web services on the internet and gathering related information. This information is essential as it details the names and parameters of each web method offered by the corresponding web service. Each web service requires a specific document that contains this essential information. This document is known as a WSDL document, which stands for web services description language.

The WSDL document for the PPSCPCRVI web service can be accessed online at the following web address: http://194.141.86.222/ppscpcrvi/PpscpcrviWebService.asmx?wsdl

Figure 6 illustrates how the WSDL document appears when viewed in a browser.

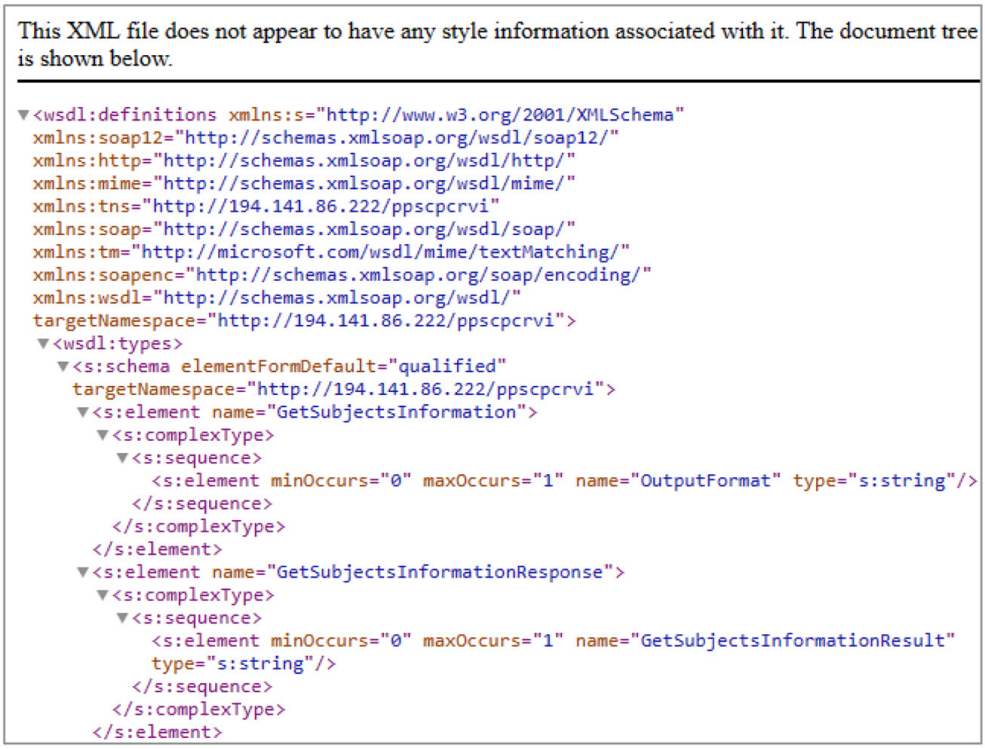

Fig. 6. The WSDL document for the PPSCPCRVI web service is displayed in a web browser

The second step in utilizing a web service is to create an intermediate class, known as a proxy class, which serves as a bridge for communication between the web functions and the calling application. Most application development environments offer tools that can automatically generate the necessary intermediate classes if the URL of the WSDL document is known or if the document is available as a local file.

The final step is to use the generated intermediate (proxy) class to call through it the web methods that the corresponding web service provides. The methods of the proxy class can be called by the application, and the methods of the proxy class themselves call the corresponding methods of the web service.

A program that uses the web functions of the Ppscpcrvi web service was built for this study. This application has been developed with the integrated development environment—Delphi part of the RAD Studio package. The proxy class (interface) definition is shown in Figure 7.

```
01 type PpscpcrviWebServiceSoap = interface(IInvokable)
02 ['{679E7379-3FBE-2334-33D6-38E89087DBCB}']
03   function GetSubjectsInformation
04     (const OutputFormat: string): string; stdcall;
05   function GetSummaryAttemptsInformation
06     (const OutputFormat: string): string; stdcall;
07 end;
08 function GetPpscpcrviWebServiceSoap(UseWSDL: Boolean;
09  Addr: string; HTTPRIO: THTTPRIO): PpscpcrviWebServiceSoap;
```

Fig. 7. The code of the proxy interface PpscpcrviWebServiceSoap declaration

The PpscpcrviWebServiceSoap class implements the IInvokable interface by adding methods corresponding to the web functions of the web service (lines 3–7). Each method contains the input parameter–OutputFormat of type string (as defined in the WSDL description). However, the parameter types are specific to the programming language being used. The GetPpscpcrviWebServiceSoap method (which is a function in this context) returns an instance of the proxy class (lines 8–9). Using this object, the application can invoke the functions declared in the proxy class within the source code. The Ppscpcrvi web service can also be integrated into applications developed using other development environments.

3 EXPERIMENTAL RESULTS

3.1 Methodology of the experiments

The aim is to investigate the different techniques for retrieving data from a database, using either web-based methods or direct queries to the database server.

An application has been created for the experiments, which implements all available methods for data retrieval from the database server. An example session showcasing the functionality of this application is presented in Figure 8. Beyond its main functions, this application includes a feature that measures the execution time of each operation.

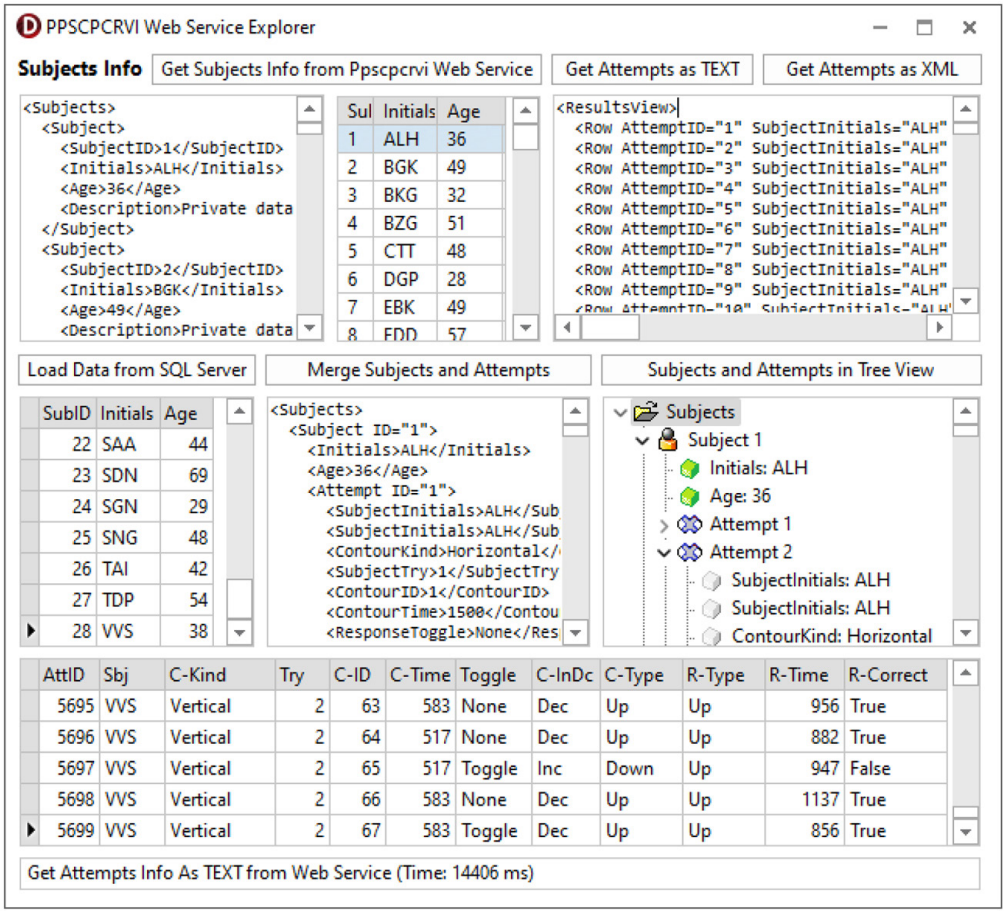

Fig. 8. The PPSCPCRVI Web Service Explorer application

This application can perform direct queries to the database server as well as call web services from the web server. The user interface of the application includes appropriate controls for visualizing data retrieved from the database server, which can be accessed either via direct queries or by calling web methods. The application visualizes the time required for each operation in its status line after the task is completed.

The experimental conditions included a computer running a 64-bit version of Win 11 Pro on an x64-based processor and the associated hardware configuration: Processor: Intel Core i5–13420H CPU at 3.40 GHz; RAM: 16GB DDR5, and remote server with 64-bit OS Win Server 2016, x64-based processor and the associated hardware configuration: Processor: Intel Xeon E5645 CPU at 2.39 GHz; RAM: 16GB DDR5. The Internet connection bandwidth is 100 Mbps (download speed: 54.28 Mbps; upload speed: 47.82 Mbsp; ping: 11 ms).

3.2 Data retrieval time based on method used

When large data sets are accessed via web services, the application's performance may suffer. This is because the web server retrieves data in blocks of records, which are subsequently sent to the calling application as separate packets. The application merges these packets and buffers them in dynamically allocated RAM, which is exclusively used by that application. The time required to retrieve all requested data

may be unacceptably long. Thus, it is important to perform experiments to evaluate how much data can be retrieved and for how long using the two methods: the first utilizing the web methods from the PPSCPCRVI web service and the second involving direct retrieval from the database server. In addition to the operations related to data extraction, we recorded execution times for two merging operations to evaluate performance. These operations involved data that had already been buffered by the application.

Table 1 provides additional information on the number of records retrieved or processed for each of the analyzed operations. The number of records is notably small only in the "Get Subjects information from PPSCPCVI web service" operation. This operation will not be analyzed because its execution time is negligible. In the remaining operations, the number of records processed is comparable, and the difference between some records is merely 28. This happens because, in operations 4, 5, and 6, the records from both the "Subjects" and "Attempts" tables are processed together.

Table 1. Analyzed operations and number of records processed

No	Operation	Records
1	Get Subjects information from PPSCPCVI web service.	28
2	Get Attempts information from PPSCPCVI web service as text.	5699
3	Get Attempts information from PPSCPCVI web service as xml.	5699
4	Loading data from Subjects and Attempts from SQL Server.	5727
5	Merge Subjects and Attempts data.	5727
6	Loading Subjects and Attempts information in tree view control.	5727

The results from the experiments are summarized in Table 2.

Table 2. Execution times (in milliseconds) for various operational tasks in 12 tests

Test	Get Subjects WS (csv)	Get Attempts WS (text)	Get Attempts WS (xml)	Load Data from SQL Server	Merge Subjects & Attempts	Subjects & Attempts in Tree View
1	78	10937	14656	246	1532	8734
2	31	11188	14985	234	1812	8765
3	47	11563	15766	219	1718	8818
4	32	11140	16265	219	1719	8752
5	47	11453	15969	203	1687	8797
6	125	11750	16063	235	1672	8746
7	79	11719	16572	234	1657	8812
8	94	11343	16063	219	1641	8875
9	93	11906	17078	313	1594	8844
10	109	12297	16625	203	1641	8735
11	46	11547	15422	204	1625	8859
12	63	11766	16640	219	1609	8797

Figure 9 graphically presents the summarized data from Table 2.

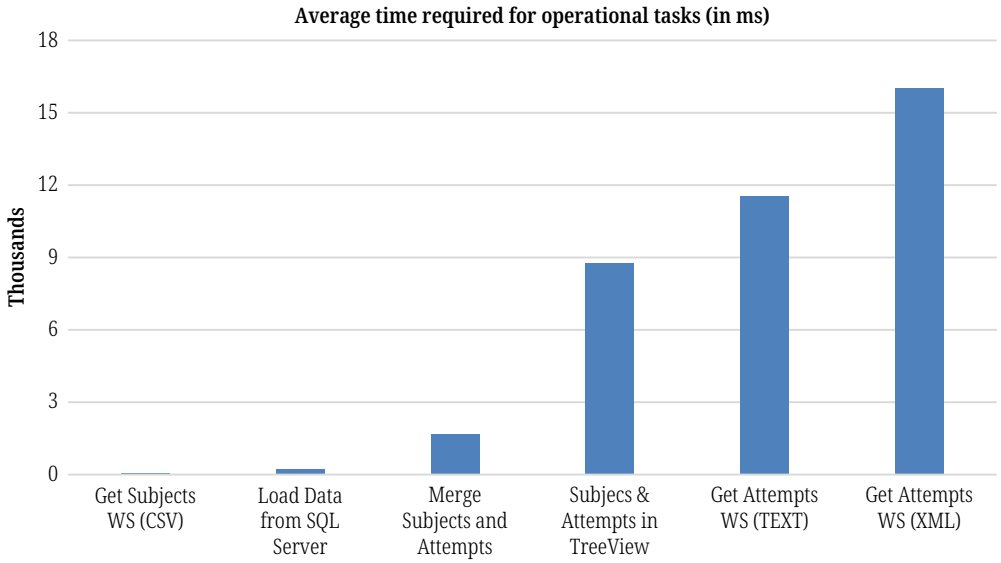

Fig. 9. Comparison of average times (in milliseconds) for performing different tasks

To more accurately estimate the time needed for executing various operations, we conducted 12 tests. This is necessary because of the multitasking mode of the operating system on which the application was run and the corresponding tests were performed. The summarized results of the various tests are shown in Table 3. These results include the minimum, maximum, and average execution times for each of the analyzed operations.

Table 3. Minimum, maximum, and average execution times for each operations

Test	Get Subjects WS (csv)	Get Attempts WS (text)	Get Attempts WS (xml)	Load Data from SQL Server	Merge Subjects & Attempts	Subjects & Attempts in Tree View
Min	31	10937	14656	203	1532	8734
Max	125	12297	17078	313	1812	8875
Avg	70	11551	16009	229	1659	8795

The results indicate that directly retrieving data from the database server (specifically during the 'Loading data from Subjects and Attempts from SQL Server' operation) is significantly faster than using a web service. Among all the operations analyzed, the two slowest are 'Get Attempts information from the PPSCPCVI web service as text' and 'Get Attempts information from the PPSCPCVI web service as XML.' Both operations use the PPSCPCVI web service to retrieve data from the 'Attempts' table. The difference in the received information is that one is in plain text format, while the other is in XML format. The slowdown is approximately 30% when using the XML data format instead of the plain text format. The execution of the 'Merge Subjects and Attempts data' operation is relatively faster than data retrieval operations based on web services. However, in this operation, the actual merging of the application data is performed on the local machine, as the data is already buffered in the application's address space and is processed locally. This is also true for the 'Loading Subjects and Attempts information in tree view control' operation.

However, this operation is much slower due to the repeated invocation of methods for constructing and presenting the data in a tree structure. In summary, the two slowest operations are related to data retrieval using web services. Their execution time is not significantly affected by the multitasking mode of the operating system on which the client application (i.e., the application that calls the web service) is running. This is evident from the diagram in Figure 10, which shows that in all tests performed, the execution time for both operations is comparable across the different tests.

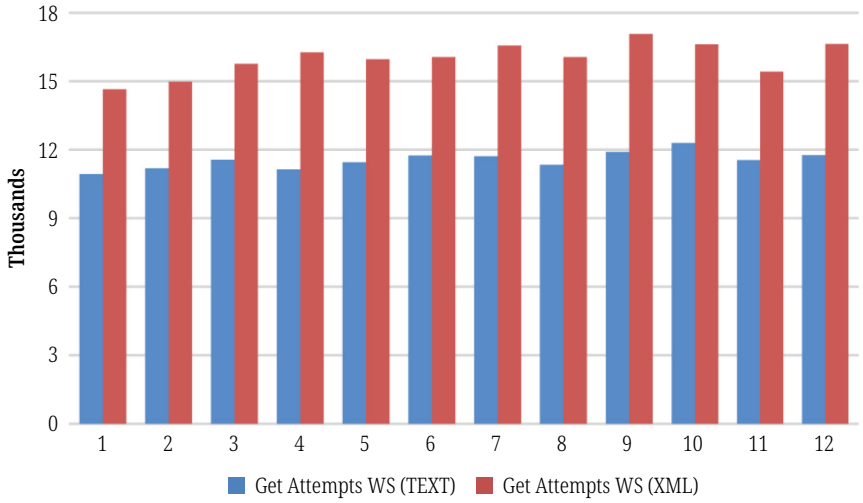

Fig. 10. Comparison of average times (in milliseconds) for retrieving attempts information from the PPSCPCVI web service in text and XML formats

Both operations take longer to execute than other operations because, when using web services, the data undergoes multiple conversions at various levels before reaching the application that uses it. In addition, web services process data by describing it in XML format. However, the self-describing syntax of the XML language requires the use of two tags (opening and closing) to describe each value. On the other hand, retrieving data via a web service allows it to be processed on different operating systems and through various applications.

4 CONCLUSION

In this paper the results of an analysis of different approaches to retrieving experimental data was presented. The different methods of processing the data and their applications were analyzed. For the study, a relational database was designed to store, analyze, and process the experimental data. The methodology and conditions for conducting the experiments were also described. For the experiments, twelve different tests were conducted, and six operations related to extracting, processing, and analyzing the experimental data were examined. The results indicate that data extraction is fastest when using a direct approach, specifically by executing a query directly on the database server. Based on the results, the two slowest operations were found to be related to data extraction using web methods. The first variant describes the data in text format, while the second uses xml. The second approach is significantly slower in data retrieval, by about 30%. In addition to data retrieval operations, data processing operations were also analyzed once the data was buffered in the address space of the client application. It was found that these operations

are performed for acceptable time—slower than direct data retrieval from the database server but faster than data retrieval using web methods.

Finally, it can be pointed out that when using the direct approach to extract data from a database server, the technologies used in both the server and the client application are interconnected and, consequently, mutually limited by each other. This disadvantage is not present when using and processing data retrieved via web services. Therefore, despite the longer execution time of operations that use web services for data access compared to those that use a direct approach to access the database server, they are essential when the goal is to allow data access (via the corresponding web methods) by applications running on different operating systems. This is a significant advantage over using a direct approach to data access, which limits the range of possible access technologies.

5 ACKNOWLEDGMENT

The reported work was supported by project № KP-06-N52/6 "Modelling post-perceptual stages of cognitive processing and conscious representations of visual information" funded by the Bulgarian Science Fund.

6 REFERENCES

[1] G. M. Afify, A. E. Bastawissy, and O. M. Hegazy, "A hybrid filtering approach for storage optimization in main-memory cloud database," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 329–337, 2015. https://doi.org/10.1016/j.eij.2015.06.007

[2] N. Wang, X. Chen, G. Song, and H. Parsaei, "An experiment scheduler and federated authentication solution for remote laboratory access," *International Journal of Online Engineering*, vol. 11, no. 3, pp. 20–26, 2015. https://doi.org/10.3991/ijoe.v11i3.4554

[3] J. Wagner *et al.*, "Carving database storage to detect and trace security breaches," *Digital Investigation*, vol. 22, pp. S127–S136, 2017. https://doi.org/10.1016/j.diin.2017.06.006

[4] J. Kim, A. Park, and S. Lee, "Recovery method of deleted records and tables from ESE database," *Digital Investigation*, vol. 18, pp. S118–S124, 2016. https://doi.org/10.1016/j.diin.2016.04.003

[5] U. Tosun, "Distributed database design: A case study," *Procedia Computer Science*, vol. 37, pp. 447–450, 2014. https://doi.org/10.1016/j.procs.2014.08.067

[6] P. Nicolaos and T. Katerina, "Simple-talking database development: Let the end-user design a relational schema by using simple words," *Computers in Human Behavior*, vol. 48, pp. 273–289, 2015. https://doi.org/10.1016/j.chb.2015.02.002

[7] V. Dimitrieski, M. Celikovic, S. Aleksic, S. Ristic, A. Alargt, and I. Lukovic, "Concepts and evaluation of the extended entity-relationship approach to database design in a multi-paradigm information system modeling tool," *Computer Languages, Systems & Structures*, vol. 44, pp. 299–318, 2015. https://doi.org/10.1016/j.cl.2015.08.011

[8] D. Botsceva, "A technological study of the linguistic competence of children with communicative disorders," St. Kliment Ohridski University Press, Sofia, 2008.

[9] S. Bergamaschi, F. Guerra, M. Interlandi, R. Trillo-Lado, and Y. Velegrakis, "Combining user and database perspective for solving keyword queries over relational databases," *Information Systems*, vol. 55, pp. 1–19, 2016. https://doi.org/10.1016/j.is.2015.07.005

[10] D. Trivedi, P. Zavarsky, and S. Butakov, "Enhancing relational database security by metadata segregation," *Procedia Computer Science*, vol. 94, pp. 453–458, 2016. https://doi.org/10.1016/j.procs.2016.08.070

[11] A. Iqbal *et al.*, "Advancing database security: A comprehensive systematic mapping study of potential challenges," *Wireless Networks*, vol. 30, pp. 6399–6426, 2024. https://doi.org/10.1007/s11276-023-03436-z

[12] M. Grahl *et al.*, "Archive WEB API: A web service for the experiment data archive of Wendelstein 7-X," *Fusion Engineering and Design*, vol. 123, pp. 1015–1019, 2017. https://doi.org/10.1016/j.fusengdes.2017.02.047

[13] V. Gharibvand *et al.*, "Cloud based manufacturing: A review of recent developments in architectures, technologies, infrastructures, platforms and associated challenges," *International Journal of Advanced Manufacturing Technology*, vol. 131, pp. 93–123, 2024. https://doi.org/10.1007/s00170-024-12989-y

[14] L. Li, M. Liu, W. Shen, and G. Cheng, "Recommending mobile services with trustworthy QoS and dynamic user preferences via FAHP and ordinal utility function," *IEEE Transactions on Mobile Computing*, vol. 19, no. 2, pp. 419–431, 2020. https://doi.org/10.1109/TMC.2019.2896239

[15] J. Z. Ruiz and C. M. Rubira, "Quality of service conflict during web service monitoring: A case study," *Electronic Notes in Theoretical Computer Science*, vol. 321, pp. 113–127, 2016. https://doi.org/10.1016/j.entcs.2016.02.007

[16] D. Redavid and S. Ferilli, "Semantic web services ingestion in a process mining framework," *Electronics*, vol. 12, no. 23, p. 4767, 2023. https://doi.org/10.3390/electronics12234767

[17] G. Alam, S. Mahmood, M. Alshayeb, M. Niazi, and S. Zafar, "Maturity model for secure software testing," *Journal of Software: Evolution and Process*, vol. 36, no. 5, p. e2593, 2024. https://doi.org/10.1002/smr.2593

[18] M. Thirumaran, P. Dhavachelvan, D. Aishwarya, and R. Shanmugapriya, "Finite state machine based access control mechanism for web service work flow management," *IERI Procedia*, vol. 4, pp. 391–397, 2013. https://doi.org/10.1016/j.ieri.2013.11.056

[19] M. Bortlik, B. Heinrich, and D. Lohninger, "Service re-selection for disruptive events in mobile environments: A heuristic technique for decision support at runtime," *Information Systems Frontiers*, vol. 26, pp. 1063–1090, 2024. https://doi.org/10.1007/s10796-023-10392-8

[20] X. Han, B. Li, K. F. Wong, and Z. Shi, "Exploiting structural similarity of log files in fault diagnosis for Web service composition," *CAAI Transactions on Intelligence Technology*, vol. 1, no. 1, pp. 61–71, 2016. https://doi.org/10.1016/j.trit.2016.03.006

[21] M. Tawfik *et al.*, "Laboratory as a service (LaaS): A novel paradigm for developing and implementing modular remote laboratories," *International Journal of Online Engineering (iJOE)*, vol. 10, no. 4, pp. 13–21, 2014. https://doi.org/10.3991/ijoe.v10i4.3654

[22] A. Eiden, T. Eickhoff, J. C. Göbel, C. Apostolov, P. Savarino, and T. Dickopf, "Data networking for industrial data analysis based on a data backbone system," in *Proceedings of the Design Society*, vol. 2, 2022, pp. 693–702. https://doi.org/10.1017/pds.2022.71

[23] F. Ghedass and F. Ben Charrada, "On the use of big data frameworks in big service management," *Journal of Software: Evolution and Process*, vol. 36, no. 7, p. e2642, 2024. https://doi.org/10.1002/smr.2642

[24] R. Kraleva, V. Kralev, and P. Koprinkova-Hristova, "Data analysis from two-choice decision tasks in visual information processing," *International Journal on Informatics Visualization*, vol. 5, pp. 187–193, 2021.

[25] R. Ratcliff and P. L. Smith, "Perceptual discrimination in static and dynamic noise: The temporal relation between perceptual encoding and decision making," *Journal of Experimental Psychology: General*, vol. 139, no. 1, pp. 70–94, 2010. https://doi.org/10.1037/a0018128

[26] P. Bayerl and H. Neumann, "Disambiguating visual motion through contextual feedback modulation," *Neural Computation*, vol. 16, no. 10, pp. 2041–2066, 2004. https://doi.org/10.1162/0899766041732404

[27] P. Koprinkova-Hristova, M. Stefanova, B. Genova, N. Bocheva, R. Kraleva, and V. Kralev, "Features extraction from human eye movements via echo state network," *Neural Computing and Applications*, vol. 32, pp. 4213–4226, 2020. https://doi.org/10.1007/s00521-019-04329-z

[28] P. D. Koprinkova-Hristova, N. Bocheva, S. Nedelcheva, and M. Stefanova, "Spike timing neural model of motion perception and decision making," *Frontiers in Computational Neuroscience*, vol. 13, p. 20, 2019. https://doi.org/10.3389/fncom.2019.00020

[29] R. Zargari Marandi, P. Madeleine, O. Omland, N. Vuillerme, and A. Samani, "Eye movement characteristics reflected fatigue development in both young and elderly individuals," *Scientific Reports*, vol. 8, 2018. https://doi.org/10.1038/s41598-018-31577-1

[30] R. Ratcliff, P. L. Smith, S. D. Brown, and G. McKoon, "Diffusion decision model: Current issues and history," *Trends in Cognitive Sciences*, vol. 20, no. 4, pp. 260–281, 2016. https://doi.org/10.1016/j.tics.2016.01.007

[31] K. X. Chiong, M. Shum, R. Webb, and R. Chen,"Combining choice and response time data: A drift-diffusion model of mobile advertisements," *Management Science*, vol. 70, no. 2, pp. 1238–1257, 2024. https://doi.org/10.1287/mnsc.2023.4738

[32] V. Kralev, R. Kraleva, and P. Koprinkova-Hristova, "Data modelling and data processing generated by human eye movements," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 5, pp. 4345–4352, 2021. https://doi.org/10.11591/ijece.v11i5.pp4345-4352

[33] R. Kraleva, V. Kralev, N. Sinyagina, P. Koprinkova-Hristova, and N. Bocheva, "Design and analysis of a relational database for behavioral experiments data processing," *International Journal of Online Engineering*, vol. 14, no. 2, pp. 117–132, 2018. https://doi.org/10.3991/ijoe.v14i02.7988

[34] V. Kralev, R. Kraleva, N. Sinyagina, P. Koprinkova-Hristova, and N. Bocheva, "An analysis of a web service based approach for experimental data sharing," *International Journal of Online Engineer*ing, vol. 14, no. 9, pp. 19–34, 2018. https://doi.org/10.3991/ijoe.v14i09.8740

[35] R. H. Kulkarni, P. Padmanabham, M. Harshe, K. K. Baseer, and P. Patil, "Investigating agile adaptation for project development," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 3, pp. 1278–1285, 2017. https://doi.org/10.11591/ijece.v7i3.pp1278-1285

[36] X. Zhang, J. Liu, B. Cao, and M. Shi, "Web service classification based on information gain theory and bidirectional long short-term memory with attention mechanism," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 13, p. e6202, 2021. https://doi.org/10.1002/cpe.6202

[37] S. Garba, R. Mohamad, and N. A. Saadon, "Web service discovery approaches for dynamic mobile environment: A systematic literature review," *International Journal of E-Services and Mobile Applications*, vol. 13, pp. 16–38, 2021. https://doi.org/10.4018/IJESMA.2021100102

[38] X. Qiao *et al.*, "A container-based approach for sharing environmental models as web services," *International Journal of Digital Earth*, vol. 14, no. 8, pp. 1067–1086, 2021. https://doi.org/10.1080/17538947.2021.1925758

7 AUTHORS

Velin Kralev is an Associate Professor of Computer Science at the South-West University, Blagoevgrad, Bulgaria. He defended his Ph.D. Thesis in 2010. His research interests include optimization problems of the graph theory and component-oriented software engineering (E-mail: velin_kralev@swu.bg).

Radoslava Kraleva is an Associate Professor of Computer Science at the South-West University, Blagoevgrad, Bulgaria. She defended her Ph.D. Thesis in 2014. Her research interests include speech recognition, mobile app development, and computer graphic. She is a reviewer of "International Journal on Advanced Science, Engineering and Information Technology" (E-mail: rady_kraleva@swu.bg).

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 2019 she has been a Professor 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 spike timing neural networks, reinforcement learning and BCI.

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.

Miroslava Stefanova is an Assistant Professor at the Institute of Neurobiology, Bulgarian Academy of Sciences. She received a PhD degree in psychophysiology in 2014 from the Institute of Neurobiology, Bulgarian Academy of Sciences. Her research interests include motion perception, human visual and aging.