BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 67, No. 3, 2019
DOI: 10.24425/bpasts.2019.129652

A wavelet-based approach for business protocol discovery of web services from log files

A. MOUDJARI*, I. KEZZOULI, H. TALBI, and A. DRAA

MISC Laboratory, Constantine University, Ali Mendjeli, Algeria

Abstract. Recently, business protocol discovery has taken more attention in the field of web services. This activity permits a better description of the web service by giving information about its dynamics. The latter is not supported by theWSDL language which concerns only the static part. The problem is that the only information available to construct the dynamic part is the set of log files saving the runtime interaction of the web service with its clients. In this paper, a new approach based on the Discrete Wavelet Transformation (DWT) is proposed to discover the business protocol of web services. The DWT allows reducing the problem space while preserving essential information. It also overcomes the problem of noise in the log files. The proposed approach has been validated using artificially-generated log files.

Key words: business protocol discovery, wavelets, log files, denoising, dimension reduction.

1. Introduction

Web services are considered as the best way to ensure interoperability between applications on the internet. A web service exchanges with its partners a set of messages respecting a standard protocol for fulfilling its goals. This allows applications to communicate easily without generating additional cost for companies. So, companies knowing the added value of web services have adopted this paradigm to achieve their goals and stay competitive in the open market. Consequently, a web service can be the result of a legacy application migration or a novel program developed as a web service. A developer is not necessarily obliged to give all the specification and documentation to describe the real behaviour of a web service. In general, the focus is on what a web service can do correctly. Other details, such as the order of exchanged messages, are omitted.

A web service is described using the WSDL language. This description concerns only the static part, i.e. the messages that could be exchanged during all possible conversations. Programmers have not to provide the order according to which these messages are exchanged. This order represents the actual behaviour of the web service that represents its dynamic part. The behaviour could be observed from the recorded log files that include the history of the service execution. So, log files are likely to be the appropriate source from which the dynamic part of the web service could be described. In fact, log files have been used in several fields and applications such as workflow mining [1], healthcare systems [2], network analysis [3], etc. They have emerged as a very useful means to record the events produced during the system runtime.

A logical way of discovering the business protocol (BP) of a web service is to build a graph which represents its dynamic part. This dynamic part is generally useful for enhancing the description of the web service. Besides, it permits reusing the web service, adding some constraints and allowing the development of adapted client applications. The result of the discovery process could be exploited by a human expert or used by a software for web services comparison or for composition aims. The presence or absence of a conversation ID (CID) in the log file would decide the best way to build the corresponding graph. If the CID is present, the problem would be simple to deal with. However, log files of real-world applications do not generally contain such an identifier, the developer is not obliged to provide an unneeded piece of information. The real problem, thus, rises in the case of absence of the CID from the log file. In this situation, the problem of BP discovering could be dealt with on the basis of strength of correlation between adjacent messages. It is a fact that noise and the absence of CID make it practically impossible to use a simple trivial algorithm to discover the BP from a given log file. Fortunately, adjacent messages are likely to occur in the same area of the log file. In this context, the present work proposes a new approach based on the use of discrete wavelets for discovering business protocols from the log-files containing the traces of their execution. Because of the lack of real-applications log files [4], the implemented approach has been tested on a set of graphs describing the behaviour of some business protocols. The graphs were used to generate different log files with different noise rates. The considered business protocols model the interaction between the web service and the clients. The obtained results have been compared to those of the recently developed LSA-based algorithm [5].

The rest of the paper is organised as follows. After this introduction, some basic concepts are given in Section 2. In Section 3, some related works are presented and commented. The proposed approach is described in detail in Section 4. Section 5 is devoted

*e-mail: moudjariabdelkader@gmail.com

Manuscript submitted 2018-05-16, revised 2018-10-25 and 2018-12-09, initially accepted for publication 2018-12-12, published in June 2019.

to the validation of the proposed approach. Finally, a conclusion and some perspectives are given in Section 6.

2. Basic concepts

The first part of this section presents the basic concepts related to business protocol discovery. The second part concerns the discrete wavelet transform (DWT) which is at the centre of our approach.

2.1. Business protocols – definitions: The definitions presented in this section are mainly taken from [6]:

- **Log file:** A log file is a set of entries. Each entry represents an event. This event is produced when the web service exchanges (sends/receives) a message with its partners. In fact, a log file keeps the runtime trace of a given application.
- **Event:** An event could be modelled as a set of attributes. The main attributes are the time stamp, sender identifier, receiver identifier, and message type. Some works suppose the existence of the conversation identifier.
- **Message:** The message is the interaction unit between the partners to fulfil a goal. Each message has a type. There are two important message types: *initial* and *final*.
- **Couple of messages:** This concept appeared the first time in [7]. The authors, there, used it to establish a connection between messages in the same set. It was used to determine the communication direction. The couple of messages represents the link between a message and its successor.
- **Conversation:** A conversation is a set of messages exchanged respecting a given order. This order goes from the initial state to a final state. Partial conversations are not taken into account.
- **Business protocol:** A business protocol (BP) is the specification of all possible conversations supported by the web service during its runtime to achieve a goal. It describes the dynamic part of a web service.

2.2. Business protocol formal definition. A business protocol is defined by the tuple: $P = (S, S_0, F, Msg, R)$ where:

- S is a finite set of states.
- S_0 is the initial state.
- $F \subseteq S$ is a set of final states. If $F = \varnothing$, P is considered as an empty protocol.
- Msg is a finite set of messages. For each message $m \in Msg$, a polarity function is defined. It has a positive value $(+)$ if the message is coming in and a negative value $(-)$ otherwise. The $(+)$ or the $(-)$ shows the orientation of the edge.
  In our context, this function is replaced by the notion of 'couple of messages' extracted from the sub log. In order to keep the polarity notion without using any function, a set of couple of messages is constructed according to a rule. This rule takes into account the message and its successor in the log files.
- $R \subseteq S \times Msg \times S$ is a finite set of transitions. Each transition (s, m, s') is composed of a state source s and another target state s'. The transition from one state to another consumes an incoming or outgoing message.

2.3. Wavelets: Wavelets have been used in many domains such as statistics [8–10], time series analysis [11–13], biological data analysis [14, 15], signal processing [16–18], image processing [19–23], speech recognition [24] and databases [25]. Recently, a new orientation that adopts the wavelets in the field of data mining [26, 27] has emerged. In our context, we are interested in the aspect of using the wavelets as a mechanism to reduce the problem's dimension. In other words, the goal of dimension reduction is to express the original dataset using some smaller set of data with or without information loss [26].

The authors in [26] gave an overview about using the wavelets to extract information from a dataset. In addition, they explained the essential properties of wavelets such as: compact support, the vanishing moments and dilating relations. Compact support guarantees the localisation of wavelets (processing region of data with wavelets does not affect the data out of this region). The vanishing moment permits to distinguish the essential information form non-essential information. The dilating relation makes the wavelet algorithm work faster. In [28], a classification of approaches using wavelets as a method for reducing the problem's dimension has been presented. Those approaches were based on a wavelet decomposition tree [29] and multiresolution signal decomposition [16]. In general, they divide recursively the dataset into two subsets each time. Fig. 1 illustrates the process. The most important coefficients used are the approximation coefficient A and the details coefficient D (see [27] for more details).

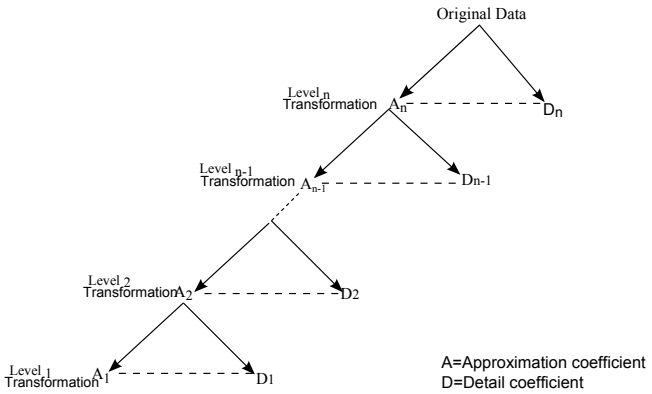

Fig. 1. Wavelets decomposition tree, adapted from [28] and [29]

3. Related work

Recently, business protocol discovery has attracted the attention of many researchers. The works in this field can be categorised into two families. The first family assumes the presence of a conversation identifier (CID) as an attribute in the events of the log files, which allows gathering messages belonging to the same conversation. So, it makes the process of discovering business protocols easy. The second family, in contrary, does not take into account this supposition, and the process of gathering messages belonging to the same conversation becomes more difficult. The question is 'how to act in order to recognise a conversation from the others?' Statistical techniques are generally used to

Table 1
Summary of some related works

Authors	Approach	Critics
Benattalah et al. 2004 [6]	Find among all event attributes those that can play a conversation identifier role (messages correlation)	The complexity increases with the number of attributes in the event
Devaurs et al. [30]	Using the notion of 'episode' to determine the limit of each conversation	Detects the temporal transition, but the problem is how to estimate the episode's beginning and end
Serrour et al. 2008 [31]	Using graph theory to detect the BP	The problem of noise is not treated
Musaraj et al. 2010 [32]	Linear regression is adopted as method to extract BP	The noise problem is not solved, and critical cases are not treated
Moudjari et al. 2014 [5]	Latent semantic analysis is used to reduce the dimension of the problem and identify conversations	The noise problem is treated but the self-loop problem is not solved
Moudjari et al. 2015 [7]	Use an adapted TF-IDF (Term Frequency – Inverse Document Frequency) measure as mechanism to find the arcs representing the BP	Both noise and self-loop problems are treated but time consumption remains high

separate such conversations.Our approach permits to discover the business protocol without any prior information about the web service being discovered. It belongs, then, to the second family, i.e. without the CID. Table 1 gives a description of some works that do not consider CID presence.

4. The proposed approach

In this section, we first give the motivation behind our choices. Afterwards, a general description of the proposed approach is presented. Then are given the details about the different steps: partitioning the log file into sub-logs, constructing the occurrence matrix, applying the wavelet transform on the occurrence matrix, computing the correlation matrix, and constructing the discovered BP graph. This process is illustrated in the chart of Fig. 2.

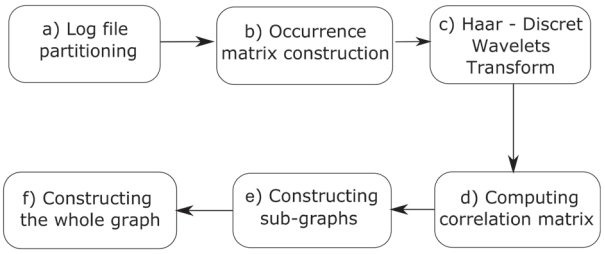

Fig. 2. The proposed approach process

4.1. Motivation. The motivation behind using couples of messages instead of single messages is to find the order of messages. In [5], the authors used single messages as correlation units, and to establish links between the messages belonging to the same conversation, they have defined two structures called micro- and macro-successors. The first structure is used to connect the messages in the same set, and the second serves to establish links between different sets. The main drawback of that approach is: its incapacity to detect self-loops in the graph and its relatively high complexity. The notion of couples of messages is then introduced to deal with the self-loop problem. The Wavelets are used, in the present work to reduce the dimension of the problem without losing the essential information and to overcome the noise problem in the log files.

4.2. The steps of the proposed algorithm. To illustrate the details of each step, the graph of Fig. 3 representing the dynamic part of a web service is used as an example. The nodes represent messages and the arcs are for the order according to which the messages are exchanged. A set of log files is generated using the up-cited graph with noise introduced. The branches are selected randomly.

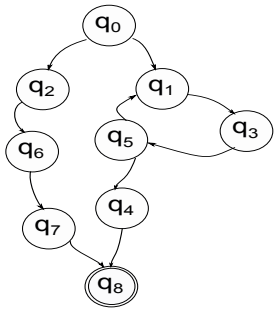

Fig. 3. A graph representing the dynamic part of a web service

4.2.1. Partitioning the log files. In this step, the log files are partitioned into sub-logs according to the attribute sender/receiver. The objective is to obtain a set of sub-logs used as input to the second step. Fig. 4 represents a log file containing a set of attributes, on top: the client address, production time of the message, name of invocation method, type of the received or sent message and the protocol used for data transfer, respectively. On the bottom of this figure, the sub log is obtained according to the client address attribute. For example, the first line in the log file of the figure gives an idea about the event in a log file ($@Ip, Timestamps$, $Invocation\_method$,

Type_of_message, *Transfer_protocol*). The most important attributes according to the related works are: *the client address, timestamps, sender/receiver and type of message*. The work in this paper focuses on finding the correlation between messages belonging to the same conversation, and getting from all the found conversations the business protocol being discovered.

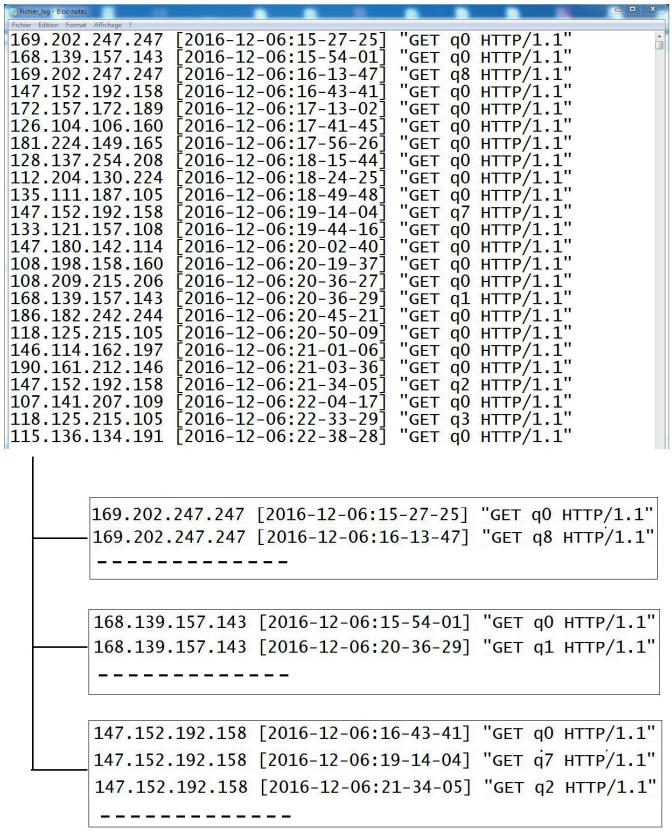

Fig. 4. A portion of a log file and its partitioning into sub-logs

4.2.2. Construction of the occurrence matrix. In this step, the occurrence matrix is built. It has as columns couples of messages and as lines clients. Both clients and couples of messages are extracted from the sub-logs. A couple represents a message and its successor in the sub-log. After establishing both lists of couples of messages and clients, the occurrence matrix is filled. A cell in this matrix represents the number of occurrences of a particular couple in a specific sub-log. To construct the couples-of-messages list, each sub-log is sorted according to the 'timestamp' attribute. A web service can lead parallel conversations with many clients at the same time. For this reason, we partition the log files into sub-logs. Each sub-log allows to separate messages ingoing or outgoing from a particular client during its interaction with a web service. Generally, the messages in the sub-log are ordered. The order could be altered in some cases, for example when the web service is strongly solicited or when the network is overloaded. In such cases, the messages could be missed or saved in the wrong order. Table 2 presents this matrix showing couples of messages as columns and client addresses as lines.

As shown in Fig. 4, the couples are constructed only using the sub-logs obtained in the previous step. In fact, we have constructed a list of couples that describe the real order of exchanging messages. We have also considered the couples relating multiple client access to the web service. Those couples have as elements the final state and initial state, (q_8, q_0) in the graph. In addition, the couples caused by noise are taken into account, for example the couple (q_1, q_5). Those couples caused by noise are extracted from the sub-log. This type of couples exists in the sub-log but is absent in the graph (no direct edges between messages).

4.2.3. Application of the wavelet reduction method. The occurrence matrix is transformed using the DWT (Discrete wavelets transform). DWT allows reducing the dimension of the occurrence matrix and eliminating noise from log files. This process is repeated until reaching a given level of details.

To get such a dimension reduction, we have proceeded in a way close to that used in [28]. At each level, a matrix of $nbl/2$ lines is computed from the nbl lines matrix. The values of the odd and even lines are combined to get a line in the new matrix. If the number of lines is odd, a new line with all values set to 0 is added. Table 3 gives the result we obtain from applying this operation on the occurrence matrix given in Table 2.

The Haar decomposition halves the size of data at each transformation level. The calculation is computationally inexpensive [28]. We have adopted a different labelling system that considers the matrix size instead of the decomposition stage. The level label is related to the number of lines in the matrix; for Level N, the matrix contains 2^N lines. For example, in Table 3, the level is 2 and the number of elements assigned to each vector is $4 = 2^2$. The combination of two successive lines may give two kinds of coefficients: approximation coefficients (C_{appx_i}) and difference coefficients (C_{diff_i}). These coefficients are calculated according to Eqs. (1).

$$C_{appx_i} = \frac{x_i + x_{i+1}}{\sqrt{2}}, \qquad C_{diff_i} = \frac{x_i - x_{i+1}}{\sqrt{2}}. \tag{1}$$

In our case, and in order to keep only the useful information, we have chosen the approximation coefficients. The difference coefficients represent the details that are not too significant in defining the business protocol. For example, Table 2 contains 8 lines, we add the value of each element of the first line to the corresponding value in the second line. The obtained value is divided by $\sqrt{2}$. We proceed with the same way for all lines.

4.2.4. Correlation matrix. Since we do not have conversation identifiers (CID), we should find a way to gather messages belonging to the same conversation. We have chosen the Pearson measure which is one of the most used correlation measures. The reduced matrix obtained in the previous step is used to compute the degree of correlation between messages. A threshold is used to gather correlated messages in common sets. In fact, when a web service interacts with its partners, it exchanges a set of messages according to a predefined order. So, messages in the same branch will occur in almost the same order in many parts of the log files, and with almost the same frequency.

Table 2
Occurrence matrix (clients/ couples)

	(q_0,q_2)	(q_2,q_6)	(q_6,q_7)	(q_7,q_8)	(q_8,q_0)	(q_8,q_1)	(q_1,q_3)	(q_3,q_5)	(q_5,q_1)	(q_5,q_4)	(q_4,q_7)	(q_0,q_1)	(q_1,q_5)
100.125.160.176	5	5	5	10	8	1	11	11	6	5	5	4	0
147.129.145.152	6	6	6	10	9	0	6	6	2	4	4	4	0
189.192.245.101	5	5	5	10	9	0	7	7	3	5	5	5	1
104.135.123.239	4	4	4	10	9	0	14	14	8	6	6	6	0
186.161.130.162	8	8	8	10	9	0	2	2	0	2	2	2	0
148.241.114.194	5	5	5	10	9	0	11	11	6	5	5	5	0
158.221.180.231	7	7	7	10	9	0	3	3	0	3	3	3	0
153.204.225.139	2	2	2	10	9	0	15	15	7	8	8	8	0

Table 3
Wavelets transformation of occurrence matrix

(q_0,q_2)	(q_2,q_6)	(q_6,q_7)	(q_7,q_8)	(q_8,q_0)	(q_8,q_1)	(q_1,q_3)	(q_3,q_5)	(q_5,q_1)	(q_5,q_4)	(q_4,q_7)	(q_0,q_1)	(q_1,q_5)
7.778	7.778	7.778	14.142	12.020	0.707	12.020	12.020	5.656	6.363	6.363	5.656	0.0
6.363	6.363	6.363	14.142	12.727	0.0	14.849	14.849	7.778	7.778	7.778	7.778	0.707
9.192	9.192	9.192	14.142	12.727	0.0	9.192	9.192	4.242	4.949	4.949	4.949	0.0
6.363	6.363	6.363	14.142	12.727	0.0	12.727	12.727	4.949	7.778	7.778	7.778	0.0

Table 4
The correlation matrix

	(q_0,q_2)	(q_2,q_6)	(q_6,q_7)	(q_7,q_8)	(q_8,q_0)	(q_8,q_1)	(q_1,q_3)	(q_3,q_5)	(q_5,q_1)	(q_5,q_4)	(q_4,q_7)	(q_0,q_1)	(q_1,q_5)
(q_0,q_2)	1.0	**1.0**	**1.0**	0.280	0.056	0.867	0.208	0.208	**0.997**	**1.0**	**1.0**	**0.997**	0.867
(q_2,q_6)	**1.0**	1.0	**1.0**	0.280	0.056	0.867	0.208	0.208	**0.997**	**1.0**	**1.0**	**0.997**	0.867
(q_6,q_7)	**1.0**	1.0	1.0	0.280	0.056	0.867	0.208	0.208	**0.997**	**1.0**	**1.0**	**0.997**	0.867
(q_7,q_8)	0.280	0.280	0.280	1.0	**0.972**	0.319	**0.996**	**0.996**	0.224	0.280	0.280	0.224	0.319
(q_8,q_0)	0.056	0.056	0.056	**0.972**	1.0	0.471	**0.987**	**0.987**	0.009	0.056	0.056	0.009	0.471
(q_8,q_1)	0.867	0.867	0.867	0.319	0.471	1.0	0.379	0.379	0.884	0.867	0.867	0.884	**1.0**
(q_1,q_3)	0.208	0.208	0.208	**0.996**	**0.987**	0.379	1.0	**1.0**	0.147	0.208	0.208	0.147	0.379
(q_3,q_5)	0.208	0.208	0.208	**0.996**	**0.987**	0.379	**1.0**	1.0	0.147	0.208	0.208	0.147	0.379
(q_5,q_1)	**0.997**	**0.997**	**0.997**	0.224	0.009	0.884	0.147	0.147	1.0	**0.997**	**0.997**	**1.0**	0.884
(q_5,q_4)	**1.0**	**1.0**	**1.0**	0.280	0.056	0.867	0.208	0.208	**0.997**	1.0	**1.0**	**0.997**	0.867
(q_4,q_7)	**1.0**	**1.0**	**1.0**	0.280	0.056	0.867	0.208	0.208	**0.997**	**1.0**	1.0	**0.997**	0.867
(q_0,q_1)	**0.997**	**0.997**	**0.997**	0.224	0.009	0.884	0.147	0.147	**1.0**	**0.997**	**0.997**	1.0	0.884
(q_1,q_5)	0.867	0.867	0.867	0.319	0.471	**1.0**	0.379	0.379	0.884	0.867	0.867	0.884	1.0

Table 4 represents a symmetric matrix which describes the correlation between all couples of messages. For example, the couple (q_0,q_2) has a value equal to 1 with the couples that follow: (q_2,q_6) and (q_6,q_7). That means these couples are correlated. This is true according to the graph of Fig. 3.

4.2.5. Construction of the graph. In this step, the whole graph representing the dynamic part of the web service is constructed. Our approach starts from a local vision represented by the links between the messages inside each set obtained in the previous step. For constructing the global vision, the sets are connected by couples of messages belonging to different sets. To do so, a temporary graph is first constructed. This graph may contain arcs incorrectly added during the discovery process. For example, arcs relating final states to the initial state will be still there. The final graph is constructed through a procedure that starts by detecting the initial state, then the final states. Afterwards, unnecessary arcs are deleted.

Detecting the initial state. Algorithm 1 describes the process proposed to find the initial state message among a list of candidate messages. The business protocol has only one initial state, it may start from any message because of noise or network overload, where not all exchanged messages are necessarily saved.

The algorithm determines among the candidate messages the one which occurs more in the log file.

Algorithm 1 Initial State Detection

1: For each sub-log, find the message that appears at the first position
2: Construct a set I that contains messages found in Step 1
3: Compute the number of occurrences of each message from I for each sub-log
4: The message having the highest occurrence value among all messages in I is considered as the initial state

Detecting the final states. Algorithm 2 is used to find the final states. In fact, according to its definition, a Business Protocol can have one or many final states.

Algorithm 2 Final States Detection

1: For each sub-log, find all messages that appear in the last position and put them in the set F
2: Count for each message from F the number of its occurrences in last position within all sub-logs
3: Compute a threshold on the basis of standard deviation of the messages existing in F
4: Keep the messages having values greater than the threshold found in Step 3

According to the graph of Fig. 3, used as example to illustrate the process of the proposed approach, there is one final state. This is reflected by the number of occurrences of a particular couple of messages (q_7, q_8). This couple occurred in the last position in sub-logs, and it has a high value among couples in the occurrence matrix (Table 2). This is due to the fact that all possible conversations must go through this path starting from the initial state and reaching the final state. However, if the business protocol has many final states, a threshold that concerns the messages in the set F is computed and used as a metric to keep only the messages that can be considered as final states.

Detecting critical arcs. The next step, after establishing both initial and final states, is to detect incorrectly added arcs. These arcs are produced during the interaction of the web service with its partners. We have two kinds of incorrect arcs to delete. The first concerns the arcs considered as noise, characterised by their low occurrence compared to other arcs. Algorithm 3 is based on this assumption and performs an analysis to detect these arcs. The other kind concerns the arcs generated when a client makes at least two conversations. These arcs link the final states to the initial state.

Algorithm 3 Critical Arc Detection

1: Calculate a threshold for the couple of messages based on the standard deviation from the DWT matrix
2: Keep in a set C only the arcs with smaller value than the threshold found in Step 1
3: Extract final states using Algorithm 2
4: Add into C each arc going from a final state to the initial state

This algorithm gives a set of couples of messages which are considered as critical arcs. They are deleted from the temporary graph to construct the final graph. These couples in our example are: (q_8, q_1), (q_1, q_5) and (q_8, q_0). Deleting the arcs produced by noise or added incorrectly allows keeping only the correct couples of messages insides the sets. From the correlation matrix of Fig. 4, the obtained sets are:

$$G_1 = \{(q_0,q_2),\ (q_2,q_6),\ (q_6,q_7),\ (q_5,q_1),\ (q_5,q_4),\ (q_4,q_7),\ (q_0,q_1)\};$$
$$G_2 = \{(q_7,q_8),\ (q1,q3),\ (q_3,q_5)\};$$

Graph construction process. In this section, the details about building the final graph are given. This operation involves two visions: a local vision and a global vision.

- **Local vision:** The process starts from a local vision inside each set (the left side of Fig. 5). The elements (messages) are extracted from the correlation matrix. The right side of the figure illustrates the operation of connecting couples of messages inside each set. In fact, the messages in the same set are strongly correlated. At this point, a preliminary image about the graph is obtained. This local vision permits to focus only on the couples of messages in each set independently from other sets. Thus, it is a stage within a gradual approach to achieve the final goal.

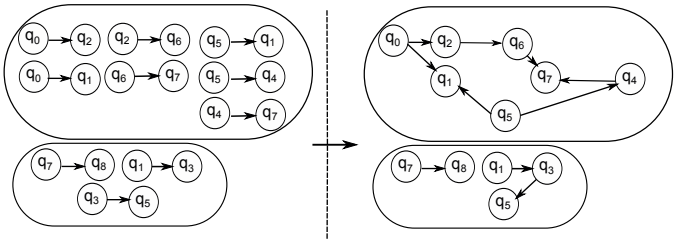

Fig. 5. Construction of sets and connecting couples inside sets

- **Global vision:** This aims to establish a connection between sets. This is performed through connecting the messages which are not located in the same set. The above part of Fig. 6 shows the common messages belonging to different sets. These messages are surrounded with discontinued lines (q_7,q_1,q_5). In the below part of the same figure, the link between messages is completed.

At the end of this process, the final graph is obtained. This operation allows building a graph that describes the behaviour of the web service during its interaction with the clients. At first glance, we can notice the similarity between the graph in Fig. 3 and the graph obtained after applying our approach (Fig. 6). In fact, the first graph does not really exist. The only information available is the log files corresponding to the web service runtime. The problem tackled in this paper is to propose

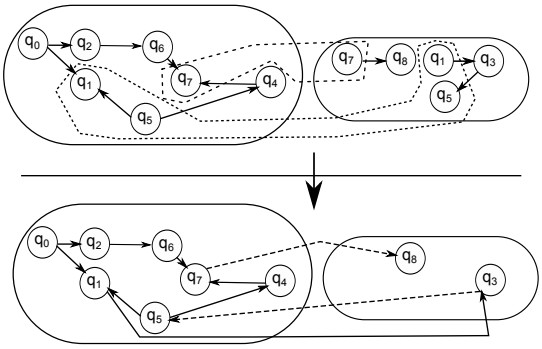

Fig. 6. Connecting sets

a method to find all possible conversations (paths in the graph which start from the initial state and reach the final states). In the next section, the details about the validation of our proposal are given.

5. Experimentation and validation

To validate our approach, the method used in [31,33] is adopted. This method makes a comparison between two graphs: the input graph and the graph discovered using a given approach. The comparison focuses on the number of discovered arcs. In other words, the degree of similarity between the two graphs. Fig. 7 illustrates this process.

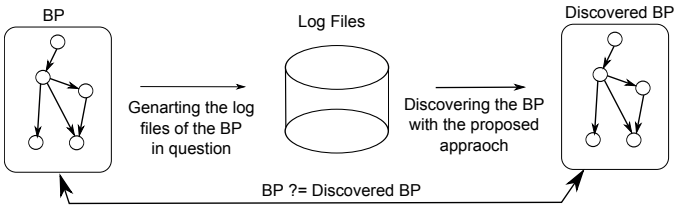

Fig. 7. The validation process

5.1. A Genuine scenario. First of all, we have taken a web service representing a real-life activity. This web service is transformed into a graph. This transformation has been done as follows. The messages are considered as states, and the arcs show the order of exchanging messages. This web service has been presented in [5] and taken from [34].

Fig. 8 represents the messages and their designation of the considered online-trade web service. The reason behind this transformation is to show the power of using a graph representation as mechanism for modelling the behaviour of the web service (runtime of a web service in terms of messages exchanged with its partners to fulfil a goal).

In addition, the graph-based representation is considered as a useful tool for the end-users to understand very well the dynamics of the web service. In fact, we start from this assumption to create graphs with different complexity levels (the number of states and edges) to validate our approach.

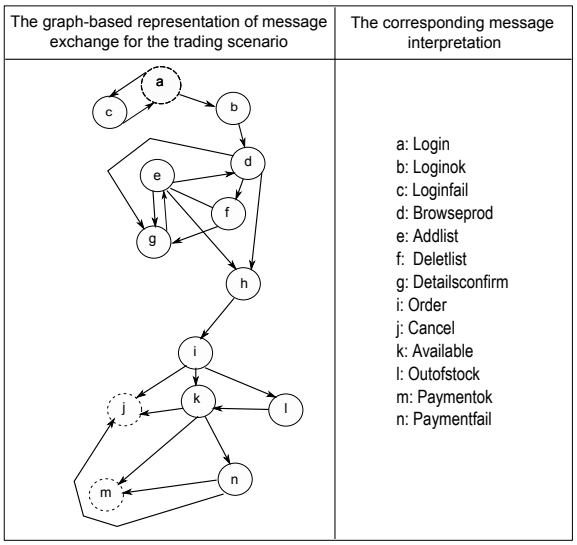

Fig. 8. Representing the online-trade web service in terms of message succession [5]

5.2. Experimental analysis. In this section, more details about validation are given. We apply the proposed approach on graphs with different complexity levels. The first graph contains 13 messages (Fig. 9), the second contains 26 messages (Fig. 10), and the third is more complex with 38 messages (Fig. 11). A measure of quality, used in [7], is adopted. This measure allows quantifying the degree of similarity between the input and the output graphs according to the validation process illustrated in Fig. 7.

$$Q = \frac{Nbr\_edges - (missing + incorrect)}{Nbr\_edges}. \qquad (2)$$

The elements of this metric (Eq. (2)) are the number of arcs correctly discovered, the number of arcs incorrectly discovered and the number of missing arcs (arcs which are not discovered by our method).

5.3. Parameters setting. In order to evaluate the impact of the parameters used in our approach, many experiments have been conducted. Some variables have been analysed:

- **Noise threshold:** Unquestionably, noise exists in real log files. It could be due to different factors. A business process could be abnormally terminated by the client. It may be due a server overload or a connection failure. In addition, an event could be totally lost or the records' writing order may be disturbed. The amount of noise depends, then, on the service itself, the clients' attitudes and the available infrastructure. In order to well simulate the real functioning of the business protocol, we have chosen to produce different amounts of noise in the generated log files. This has been done through omitting messages, saving messages in the wrong order, starting conversations from arbitrary non initial states and terminating conversations before attaining one of the final states. This parameter affects directly the number of events that will be created in the log file.

- **Wavelet details level:** For testing purposes different details degrees concerning the wavelets considered. Noisy information is likely to be preserved with high wavelet detail value and will be gradually eliminated when this value decreases.
- **The number of events in the log files:** The length of the log file has an influence on the business process discovery. On the one hand, a short log file will make it difficult to capture all the possible scenarios. On the other hand, a long l og file will increase the execution time.
- **The correlation threshold value:** It is the threshold beyond which pairs of messages are considered belonging to the same sub-group. A near-to-one correlation threshold could disperse actual sub-groups because some couples of messages would be considered as noise, while a small threshold value could include into a sub-group noise arcs.

The experimentation process starts by the generation of the log files based on the chosen graph. In our work, this number depends on the number of clients that interact with the web service and the quantity of noise. Different noise rates have been considered when generating the log files. Table 5 summarises the test parameters. The first column of the table describes the number of clients to be considered. The second is the level of noise, introduced randomly. For each experiment, we take into account the number of events produced in the log file. There are also the degree of wavelet details and the correlation threshold. These parameters are applied for each graph.

Table 5
Overview of parameters used in the validation phase

Number of clients	Noise threshold	Number of events in the log file	Wavelet details	Correlation threshold
			8	
[100..10000]	1%	$f(Noise, number\ of\ clients, Input\_graph)$	6	0.75
	5%		4	0.85
	10%		3	0.90
			2	

5.4. Discussion. Table 6 shows the results obtained for each experiment applied on the Graph (a). We give here the number of deleted arcs, the number of missing arcs (not discovered arcs), the number of arcs added incorrectly, and the rate of success, according to Eq. (2).

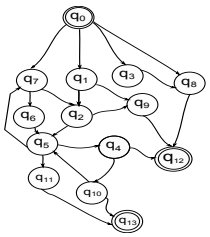

Fig. 9. The simple graph (a)

Finally, our procedure of discovering business protocols is executed to find the correct arcs that exist in the input graph.

Table 6
Graph (a) with 100 clients, 10% noise rate and 4902 events

Wavelet details	Correlation Threshold	Deleted edges	Missing edges	Incorrect edges	Success rate
8		16	22	0	0.0%
6		11	7	0	68.18%
4	0.75	11	1	0	95.45%
3		11	0	0	100.0%
2		11	0	0	100.0%
8		16	22	0	0.0%
6		11	16	0	27.27%
4	0.85	11	14	0	36.27%
3		11	1	0	95.45%
2		11	0	0	100.0%
8		16	22	0	0.0%
6		11	16	0	27.27%
4	0.9	11	14	0	36.36%
3		11	2	0	90.90%
2		11	0	0	100.0%

This procedure is repeated with different values assigned to the correlation threshold: 0.75, 0.85 and 0.90. We can see clearly the influence of the wavelet details on the obtained results. For the value 8, for example, when the number of events is small, our method was not able to distinguish between the correct arcs and those produced by noise. A clear improvement is obtained when we reduce the level of details.

Table 7 shows the results obtained when the procedure is applied for Graph (a) by changing the values of client number (5000) and noise rate (1%). This leads to produce a high number of events in the log files. Consequently, this has a positive influence on the quality of the solution. This result is confirmed in Table 8, where we can see a little growth in the deleted edge number due to the noise rate.

Table 7
Graph (a) with 5000 clients, 1% noise rate and 242900 events

Wavelets details	Correlation Threshold	Deleted edges	Missing edges	Incorrect edges	Success rate
8		2	3	0	86.36%
6		2	0	0	100.0%
4	0.75	2	0	0	100.0%
3		2	0	0	100.0%
2		2	0	0	100.0%
8		2	15	0	31.81%
6		2	2	0	90.90%
4	0.85	2	0	0	100.0%
3		2	0	0	100.0%
2		2	0	0	100.0%
8		2	16	0	27.27%
6		2	5	0	77.27%
4	0.9	2	2	0	90.90%
3		2	0	0	100.0%
		2	0	0	100.0%

Table 8
Graph (a) with 5000 clients, 5% noise rate and 242437 events

Wavelet details	Correlation Threshold	Deleted edges	Missing edges	Incorrect edges	Success rate
8		4	3	0	86.36%
6		4	0	0	100.0%
4	0.75	4	1	0	95.45%
3		4	0	0	100%
2		4	0	0	100%
8		4	16	0	27.27%
6		4	2	0	90.90%
4	0.85	4	1	0	95.45%
3		4	0	0	100%
2		4	0	0	100%
8		4	16	0	27.27%
6		4	2	0	90.90%
4	0.9	4	2	0	90.90%
3		4	0	0	100%
2		4	0	0	100%

Another series of tests has been applied with Graph (b) in order to evaluate and to perform an analysis about the influence of the parameters on the quality of the discovery process. Table 9

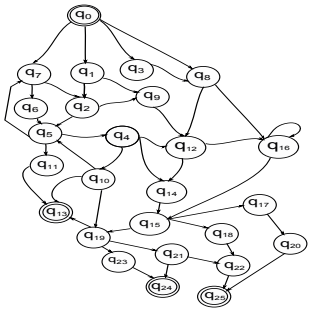

Fig. 10. The medium graph (b)

Table 9
Graph (b) with 10000 clients and 5% noise rate and 463096 events

Wavelet details	Correlation Threshold	Deleted edges	Missing edges	Incorrect edges	Success rate
8		5	14	0	67.44%
6		5	2	0	95.34%
4	0.75	5	0	0	100%
3		5	0	0	100%
2		5	0	0	100%
8		5	25	0	41.86%
6		5	5	0	88.37%
4	0.85	5	3	0	93.02%
3		5	0	0	100%
2		5	0	0	100%
8		5	27	0	37.20%
6		5	9	0	79.06%
4	0.9	5	6	0	86.04%
3		5	0	0	100%
2		5	0	0	100%

gives an overview of the experimentations with 5000 clients and a noise ratio equal to 5%. We note here the influence of the correlation threshold and the detail level on filtering the arcs. With high values for these two parameters, the algorithm was not able to distinguish between correct arcs with low occurrence and those produced by noise.

Table 10 describes another experimentation applied using Graph (b) with 10000 clients and a 10% noise ratio. We note that the number of missing arcs goes up with increasing correlation rates. The column 'missing edges' shows this deduction. The missing edges taking the values 9, 19 and 23 respectively with the threshold correlation values 0.75, 0.85 and 0.90, with a wavelet details equal to 8.

Table 10
Graph (b) with 10000 clients, 10% noise rate and 926604 events

Wavelet details	Correlation Threshold	Deleted edges	Missing edges	Incorrect edges	Success rate
8		8	9	0	79.06%
6		8	1	0	97.67%
4	0.75	8	2	0	95.34%
3		8	0	0	100.0%
2		8	0	0	100.0%
8		8	19	0	55.81%
6		8	4	0	90.69%
4	0.85	8	4	0	90.69%
3		8	0	0	100.0%
2		8	0	0	100.0%
8		8	23	0	46.51%
6		8	5	0	88.37%
4	0.9	8	6	0	86.04%
3		8	0	0	100.0%
2		8	0	0	100.0%

Tables 11 and 12 describe the results obtained from Graph (c). The remarks that we have made through the previous experiments are confirmed in these tables. Also, another aspect re-

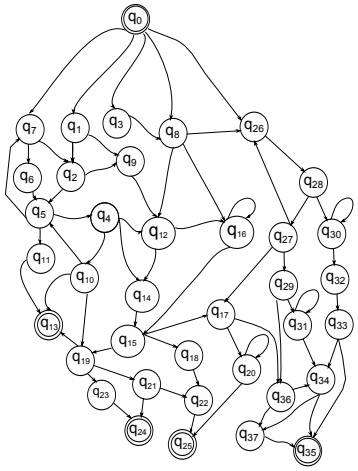

Fig. 11. The complex graph (c)

Table 11
Graph (c) with 10000 clients, noise 5% noise rate and 891509 events

Wavelet details	Correlation Threshold	Deleted edges	Missing edges	Incorrect edges	Success rate
8	0.75	6	16	0	74.19%
6		6	1	0	98.38%
4		6	2	0	96.77%
3		6	0	1	98.38%
2		6	0	1	98.38%
8	0.85	6	31	0	50.0%
6		6	3	0	95.16%
6		3	0	0	95.16%
3		6	0	1	98.38%
2		6	0	1	98.38%
8	0.9	6	37	0	40.32%
6		6	9	0	85.48%
4		6	4	0	93.54%
3		6	0	1	98.38%
2		6	0	1	98.38%

Table 12
Graph (c) with 10000 clients and 10% noise rate and 890468 events

Wavelet details	Correlation Threshold	Deleted edges	Missing edges	Incorrect edges	Success rate
8	0.75	7	15	0	75.80%
6		7	1	0	98.38%
4		7	0	1	98.38%
3		7	0	2	96.77%
2		7	0	2	96.77%
8	0.85	7	34	0	45.16%
6		7	3	0	95.16%
4		7	0	1	98.38%
3		7	0	2	96.77%
2		7	0	2	96.77%
8	0.9	7	39	0	37.09%
6		7	6	0	90.32%
4		7	5	0	91.93%
3		7	0	2	96.77%
2		7	0	2	96.77%

vealed by the two tables is the direct influence of the number of events on the number of arcs added incorrectly. This is clearly shown in the corresponding column in both tables. Through the series of experiments carried out, it can be concluded that the number of events produced in the log files is closely related to two factors: the noise ratio and the number of clients. For this reason, we have taken into account this relationship and different values have been tested. Besides, the level of details of wavelets is playing a crucial role in keeping useful information and permitting to overcome the noise problem. This parameter has been tested with many values. We notice that in the case of log files with low number of events and high level of details, the process of discovering business protocols fails as in the case of Graph (a) with 100 clients and detail level equal to 8.

5.5. Comparative study. In order to evaluate the quality of the proposed approach, a comparative study has been led and the obtained results are discussed in the current section. In fact, there are different approaches dealing with the same problem, as cited in Section 3. Among those approaches, we have selected that based on latent semantic analysis (LSA) [5] which is, as mentioned before, a recently developed method for processing log files without CID. It has been proved efficient enough and it has in common with our: (1) the global architecture; (2) the use of Pearson measure to determine correlation; and (3) gradual construction of the graph.

We have done a set of experiments on log files generated from the graphs given above according to the parameters in Table 5. We have considered two additional noise rates: 20% and 50%. The comparison does not focus only on the success rate, but also on the number of the obtained sub-groups and the number of deleted and missing arcs. To perform this comparison, we have considered the best parameters of our approach, i.e. 2 for the wavelet details level and 0.85 for the correlation threshold.

Tables 13, 14 and 15 show the experiments results that have been obtained using the graphs (a), (b) and (c), respectively. We can deduce that our method gives better results in term of success rate whatever the noise rate is. It could be noticed that our method remains efficient when the quantity of noise increases. The superiority of the proposed approach is more significant for high noise rates.

Table 13
Comparison between the proposed wavelet-based approach and LSA [5] on Graph (a) with 1000 clients

Noise	Success rate		Sub-groups		Deleted arcs		Incorrect arcs	
	Wav	LSA	Wav	LSA	Wav	LSA	Wav	LSA
1%	95.45%	95.45%	2	4	2	2	0	1
5%	95.45%	81.81%	2	6	5	2	0	4
10%	95.45%	77.27%	4	4	6	2	0	5
20%	90.90%	68.18%	4	6	9	3	0	7
50%	90.90%	59.09%	2	8	10	2	0	9

Table 14
Comparison between the proposed wavelet-based approach and LSA [5] on Graph (b) with 1000 clients

Noise	Success rate		Sub-groups		Deleted arcs		Incorrect arcs	
	Wav	LSA	Wav	LSA	Wav	LSA	Wav	LSA
1%	95.34%	100.00%	4	9	3	3	0	0
5%	97.67%	95.34%	2	10	4	3	0	2
10%	95.34%	90.69%	4	11	6	3	0	4
20%	93.02%	81.39%	3	9	10	3	0	8
50%	88.37%	74.41%	4	12	13	3	0	11

The success rate is a good metric for evaluating a method by comparing the output graph to the input graph, representing the business protocol. Of course, this supposes the availability of the input graph. In the contrary case, we have to consider a metric independent from the input graph. The number of obtained sub-groups is a good candidate for this aim since it gives an idea

Table 15
Comparison between the proposed wavelet-based approach and LSA [5] on Graph (c) with 1000 clients

Noise	Success rate		Sub-groups		Deleted arcs		Incorrect arcs	
	Wav	LSA	Wav	LSA	Wav	LSA	Wav	LSA
1%	98.38%	98.38%	9	9	4	4	0	1
5%	95.16%	96.77%	5	9	5	4	0	2
10%	93.54%	91.93%	4	8	8	4	0	5
20%	95.16%	88.70%	6	11	10	4	0	7
50%	96.77%	82.25%	4	10	14	4	0	11

on the extent to which the protocol has been discovered, i.e. to what extent the single messages have been linked together according to the functionality of the web service. A sub-group gives a local vision of the process. Constructing the whole graph representing the dynamic part of the web service relies on the obtained sub-graphs. The smaller the number of sub-groups is, the easier the construction operation of the whole graph will be. Columns 'sub-group Wav' and 'sub-group LSA' in the tables reflect the corresponding values for each method.

It could be noticed that the number of obtained sub-groups when applying the wavelet-based approach is smaller than that observed when applying the LSA-based approach. This means that our approach is better in grouping linked messages due to its robustness against noise. Another metric that we have considered in comparison, when the input graph is available, is the number of Incorrect arcs, i.e. the arcs added erroneously to the graph. Here also, one can notice the superiority of our approach that very rarely adds such incorrect arcs. In fact, the LSA-based method, unlike the wavelet-based one, often fails to distinguish between arcs produced because of noise and the actual arcs having low frequency occurrence.

The column 'deleted arcs' denotes the number of arcs considered as critical arcs linking final states to the initial state, or those caused by noise. We can conclude that our method behaves well and can eliminate those two kinds of arcs. It is clear that when the noise rate increases, the number of arcs that our method deletes increases because of its aptitude to keep useful information and eliminate noise.

6. Conclusion

In this paper, a new method for discovering business protocols from log files without any prior information is proposed. This method is based on the use of the discrete wavelet transform. It allows to manipulate a high number of events and efficiently treat the problem of noise.

The approach starts by dividing the log file into sub logs, in order to build the occurrence matrix which represents the relation between the couples of messages and the clients. The DWT is then applied to reduce the dimension of the occurrence matrix, keeping only useful information. To get the correlation between couples of messages, the Pearson measure is used as a similarity measure. Couples of messages with high correlation are gathered into the same sets. These sets are first used to define the sub-graphs, and the messages that belong to different sets allow connecting these sub-graphs. Finally, the whole graph that represents the dynamic part of the web service is built from the sub-graphs. Our method has been able to distinguish between the correct arcs with low occurrence rate and those produced by noise. This has a positive impact on the obtained results.

The proposed approach has been validated using synthetic log files. Different configurations have been tested by taking into account the presence of noise and incompleteness of data in the log files. The conducted tests with almost one million-record log files have made in evidence the ability of the proposed approach to model the dynamic part of the web service with a success rate near to 100%. The approach deals well with the noise-related problems such as missing edges, incorrect arcs and deleted edges. The comparative study conducted against the recently developed LSA-based method has proved the superiority of the new approach, especially when noise rate increases.

As future work, we plan to extend the algorithm to deal with higher numbers of events in the log files. Also, we want to use a big data framework to check the scalability of the proposed approach.

References

[1] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: discovering process models from event logs", *Knowledge and Data Engineering, IEEE Transactions on*, 16, 1128–1142, (2004).

[2] E. Helm and F. Paster, "First steps towards process mining in distributed health information systems", *International Journal of Electronics and Telecommunications* 61 (2), 137–142 (2015).

[3] N. Athavale, S. Deshpande, V. Chaudhary, J. Chavan, and S. Barde, "Framework for threat analysis and attack modelling of network security protocols", *International Journal of Synthetic Emotions (IJSE)* 8 (2), 62–75 (2017).

[4] L. Berti-Equille and M. Herschel, *The 7th International Workshop on Quality in Databases (QDB 2009) in conjunction with VLDB 2009*, Aug. 2009.

[5] A. Moudjari, S. Chikhi, and H. Kheddouci, "Latent semantic analysis for business protocol discovery using log files", *International Journal of Web Engineering and Technology* 9 (4), 365–396 (2014).

[6] B. Benatallah, F. Casati, and F. Toumani, "Analysis and management of web service protocols", in *Conceptual Modeling–ER 2004*, 524–541, Springer, 2004.

[7] A. Moudjari, S. Chikhi, and A. Draa, "Business protocol discovery from log files using a tf-idf-based technique", in *2015 Seventh International Conference on Ubiquitous and Future Networks*, 651–656, IEEE, 2015.

[8] F. Abramovich, T.C. Bailey, and T. Sapatinas, "Wavelet analysis and its statistical applications", *Journal of the Royal Statistical Society: Series D (The Statistician)* 49 (1), 1–29 (2000).

[9] A. Antoniadis, "Wavelets in statistics: A review", *Journal of the Italian Statistical Society* 6 (2), 97–130 (1997).

[10] T. Ogden, *Essential wavelets for statistical applications and data analysis*, Springer Science & Business Media, 2012.

[11] G.P. Nason and R. Von Sachs, "Wavelets in time-series analysis", *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 357 (1760), 2511–2526 (1999).

[12] D.B. Percival and A.T. Walden, *Wavelet methods for time series analysis*, 4, Cambridge University Press, 2006.

[13] P.A. Morettin, "From fourier to wavelet analysis of time series", in *COMPSTAT*, 111–122, Springer, 1996.

[14] A. Aldroubi and M. Unser, *Wavelets in medicine and biology*, CRC Press, 1996.

[15] P. Lio, "Wavelets in bioinformatics and computational biology: state of art and perspectives", *Bioinformatics* 19 (1) 2–9 (2003).

[16] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1999.

[17] P.P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial", *Proceedings of the IEEE* 78 (1), 56–93 (1990).

[18] A. Majkowski, M. Kołodziej, and R.J. Rak, "Joint time-frequency and wavelet analysis-an introduction", *Metrology and Measurement Systems* 21 (4), 741–758 (2014).

[19] L. Prasad and S. Iyengar, *Wavelet analysis with applications to image processing*, CRC Press, 1997.

[20] P. Meerwald and A. Uhl, "Survey of wavelet-domain watermarking algorithms", in *Photonics West 2001-Electronic Imaging*) 505–516, International Society for Optics and Photonics, 2001.

[21] C.E. Jacobs, A. Finkelstein, and D.H. Salesin, "Fast multiresolution image querying", in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 277–286, ACM, 1995.

[22] Y. Yang, D.S. Park, S. Huang, and N. Rao, "Medical image fusion via an effective wavelet-based approach", *EURASIP Journal on Advances in Signal Processing*, 2010 (1), 1 (2010).

[23] A. Stankiewicz, T. Marciniak, A. Dąbrowski, M. Stopa, P. Rakowicz, and E. Marciniak, "Denoising methods for improving automatic segmentation in oct images of human eye", *Bulletin of the Polish Academy of Sciences Technical Sciences* 65 (1), 71–78 (2017).

[24] S.A. Majeed, H. Husain, and S.A. Samad, "Phase autocorrelation bark wavelet transform (pacwt) features for robust speech recognition", *Archives of Acoustics* 40 (1), 25–31 (2015).

[25] D.A. Keim and M. Heczko, *Wavelets and their applications in databases*, Bibliothek der Universität Konstanz, 2001.

[26] T. Li, Q. Li, S. Zhu, and M. Ogihara, "A survey on wavelet applications in data mining", *ACM SIGKDD Explorations Newsletter* 4 (2), 49–68 (2002).

[27] P. Chaovalit, A. Gangopadhyay, G. Karabatis, and Z. Chen, "Discrete wavelet transform-based time series analysis and mining", *ACM Computing Surveys (CSUR)* 43 (2), 6 (2011).

[28] S. Russell and V. Yoon, "Applications of wavelet data reduction in a recommender system", *Expert Systems with Applications* 34 (4), 2316–2325 (2008).

[29] M. Misiti, Y. Misiti, G. Oppenheim, and J. P. Michel, *Wavelet toolbox: for use with MATLAB*. The Math Works, Inc., 1996.

[30] D. Devaurs, F. De Marchi, and M. S. Hacid, "Caractérisation des transitions temporisées dans les logs de conversation de services web", *Revue des Nouvelles Technologies de l'Information* E (9), 45–56 (2007).

[31] B. Serrour, D. P. Gasparotto, H. Kheddouci, and B. Benatallah, "Message correlation and business protocol discovery in service interaction logs", in *Advanced information systems engineering*, 405–419, Springer, 2008.

[32] K. Musaraj, T. Yoshida, F. Daniel, M.-S. Hacid, F. Casati, and B. Benatallah, "Message correlation and web service protocol mining from inaccurate logs", in *Web Services (ICWS), 2010 IEEE International Conference on*, 259–266, IEEE, 2010.

[33] W.M. Van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J. Weijters, "Workflow mining: A survey of issues and approaches", *Data & Knowledge Engineering* 47 (2), 237–267 (2003).

[34] K. Musaraj, *Extraction automatique de protocoles de communication pour la composition de services Web*. PhD thesis, Lyon University, 2010.