Message Correlation and Web Service Protocol Mining from Inaccurate Logs

Kreshnik Musaraj
Universite de Lyon
LIRIS, UMR CNRS 5205
8 Boulevard Niels Bohr
69622 Villeurbanne, France
Email: kmusaraj@liris.cnrs.fr

Tetsuya Yoshida
Graduate School of Information
Science and Technology
Hokkaido University
Sapporo, Hokkaido 060-0814, Japan
Email: yoshida@meme.hokudai.ac.jp

Florian Daniel
Department of Information Engineering
and Computer Science
University of Trento
Via Sommarive, 14 - 38123, Trento, Italy
Email: daniel@disi.unitn.it

Mohand-Said Hacid
Universite de Lyon
LIRIS, UMR CNRS 5205
8 Boulevard Niels Bohr
69622 Villeurbanne, France
Email: mohand-said.hacid@univ-lyon1.fr

Fabio Casati
Department of Information Engineering
and Computer Science
University of Trento
Via Sommarive, 14 - 38123, Trento, Italy
Email: casati@disi.unitn.it

Boualem Benatallah
School of Computer
Science and Engineering
University of New South Wales
Sydney NSW 2052, Australia
Email: boualem@cse.unsw.edu.au

***Abstract*—Business process management, service-oriented architectures and software back-engineering heavily rely on the fundamental processes of mining of processes and web service business protocols from log files. Model extraction and mining aim at the (re)discovery of the behavior of a running model implementation using solely its interaction and activity traces, and no a priori information on the target model. This paper presents an approach for correlating messages and extracting the business protocol of a web service in the realistic scenario in which correlation information is entirely absent from interaction and activity logs. Correlation is achieved through deterministic computations that result in an extremely efficient method whose extensive experiments have shown its solid reliability, robustness when dealing with complex structures, and very high performance and scalability. This approach and the underlying algorithms extend what is actually possible to achieve in the web service business protocol mining domain using incomplete and noisy data logs, and opens new horizons in back-engineering of web services. The theoretical and experimental results clearly show the leap forward achieved herein.**

I. Introduction

Workflow management systems (WFMSs) and service-oriented architectures (SOAs) are continuously and steadily relying on mining techniques for achieving a broad range of objectives. In this paper we extend the usage of *process mining* described in [23] to *web service business protocol mining* (WSPM). This is because the process of distilling a structured model description from a set of real executions concerns not only business processes and workflows, but also business protocols [3] of Web services, service-oriented architectures (SOAs) etc.

WSPM is useful for many reasons. First of all, it could be used as a tool to find out how people and/or procedures really work. Second, process mining could be used for *Divergence analysis* (DA), i.e., studying the differences between the actual implementations with the predefined behavioral model. DA is a more generic process which encompasses delta analysis [23] as a particular case, and also addresses the differences between business protocols, SOA software, models based on finite-state machines (FSM),... Applications of log mining for service protocol extraction include: post-mortem monitoring, checking the equivalence between the descriptive or prescriptive model and the actual implementation. The last application is mandatory for critical systems before deploying them online or in business platforms. We also can mention obtaining the descriptive or prescriptive model if it doesn't exist, checking for security flaws, and verifying that constraints in the execution flows are satisfied.

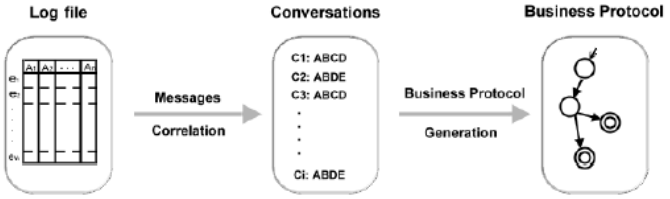

Fig. 1. The general process of business protocol discovery [22].

Yet, all the existing tools and approaches addressing mining in WFMSs and SOAs (ProM [24], Process spaceship [18] etc.) have to deal with data correlation problems which arise from the very nature of log data which tends not to fit the expectations of mining algorithms. Almost all existing approaches that address process and business protocol mining [21], [23] assume that it is possible to record data such that (i) each logged entity refers to a message or event (i.e. a unique state in the web service, or a well-defined step in the workflow), (ii) each message or event refers to a

case (i.e. a web service instance, workflow, or process), (iii) messages are totally ordered (i.e., in the log these messages are recorded sequentially), and (iv) logs contain all the information needed to correlate them to a given instance sequence. The logs considered in this paper do not obey anymore to the requirement of containing sufficient information as in [17], [19], [21], [23].

Moreover, since logs are often incomplete, uncertain and contain errors, then the first difficulty of protocol mining and message correlation approaches is the detection of these imperfections in logs. The most important factors that need to be addressed are *log incompleteness* and *noise errors*. Examples of log incompleteness are: (i) missing messages (messages are not recorded in the log), (ii) swapping messages (messages are recorded in a different order that does not reflect the real one during execution), and (iii) partial conversations (i.e. the conversation is terminated before completion). Noise errors include a large set of causes that introduce errors in the log during all the steps of the service execution and logging mechanism (hardware and failure, network problems etc.).

In this context we present the ***delta*** algorithm for correlation of messages into conversations, and service protocol discovery.

The contributions of this paper can be listed as follows:

1) Correlation of conversations with minimal assumptions on log data.
2) Algorithm capable of recognizing complex structures such as ramifications, simple and composite loops.
3) Proposal of combining statistical computations in a deterministic result that discovers the class of discoverable behavior models.

Experiments are conducted for performance and noise-proof robustness. The $delta$ algorithm aims at (i) providing the starting point for every single log-based mining method: correctly correlated and instance-sequenced data, and (ii) allowing the easy and immediate reconstruction of the model that visually displays the web service business protocol (WSBP).

The remainder of this paper is organized as follows. In Section II we introduce the required preliminaries. Section III studies the relations between messages that the $delta$ algorithm is capable of detecting in logs for achieving correlation. Based on these results, different versions of the algorithm for correlation and WSBP mining are presented. The quality and capabilities of this algorithm are supported and enhanced by the fact that it is able to rediscover correlation under very severe and limiting conditions. The paper continues in Section IV with a running example that illustrates some important aspects of the approach Section V presents the experimental results. Section VI deals with a survey on related work and we conclude in Section VII.

II. Preliminaries

A. Notations and definitions

Let Msg be a set of *message labels*. A *message type* will refer to the label of a message. A *message occurrence* is a couple $M = (m, t)$, where $m \in Msg$ is the message type, and $t \in \mathbb{R}_+$ is the timestamp of the message (denoted $M.t$). A *message log* (ML) is a collection of entries $e = (MID, m, s, r, c, t)$, where MID is the message unique identifier, m is the message type, s and r denote the sender and the receiver of message m, c the content of the message, and t is the corresponding timestamp. The timestamps are local, in the sense that no global clock is needed. If x is a message type, we will denote by $\bar{x}$ the number of occurrences of x recorded in a message log ML. An *occurrence log* (OL) is an array in which each column designates a message type, and the corresponding row value provides the number of occurrences found for that message type in a message log ML. In other words, every message log ML_i is represented as a single line in the occurrence log OL. A *conversation* is a sequence of message occurrences $C = \langle M_1, M_2, \ldots, M_n \rangle$, where $n \in \mathbb{N}^*$, and $M_1.t < M_2.t < \cdots < M_n.t$. Each web service client exchanges a precise sequence of message occurrences (i.e. a conversation) during the interaction with the web service provider. A *conversation log file* L is a multi-set of conversations.

Definition 1 A **business protocol** (BP) [4] is a tuple $P = (S, s0, \mathcal{F}, Msg, \mathcal{R})$ which consists of the following elements:
1. S is a finite set of states.
2. $s_0 \in S$ is the initial state.
3. $\mathcal{F} \subseteq S$ is a set of final states. If $\mathcal{F} = \emptyset$, then P is said to be an empty protocol.
4. Msg is a finite set of messages. For each message $m \in Msg$, a function $Polarity(P, m)$ is defined which is positive (+) if m is an input message in P and negative (-) if m is an output message in P. The notation m(+) (respectively, m(-)) is used to denote the polarity of a message m.
5. a finite set $\mathcal{R} \subseteq S^2 \times Msg$ of transitions. Each transition (s, s', m) identifies a source state s, a target state s' and either an input or an output message m that is either consumed or produced during this transition.

BP were specifically introduced to model the internal behavior of web services.

B. Linear regression

During the runtime of a WS, a conversation of messages is exchanged between the client and the service provider. The business protocol visualizes all the possible conversations that are allowed by this WS. When considering the protocol of a WS during runtime, one observes that message occurrences do not appear or disappear in a chaotic way. The first intuition is that the numbers of message occurrences for different message types are correlated between each other. This intuition leads to another one: that there is a law to which obeys the appearance of message occurrences in the log. For example, if the loop-free conversation sequence $\langle A, B, C, D \rangle$ is followed three times, then one naturally expects to find three occurrences for each message of this conversation.

The idea in this paper is to use linear regression methods to derive the equations that describe the relationships which exist between the numbers of different message occurrences. This is beneficial for many reasons. First, in order to achieve the

correlations between messages, the linear regression method requires only the number of occurrences for the given messages. To extract the full protocol after the correlation step, the approach needs only the recorded timestamps of message occurrences. Thus, an approach based on these intuitions and also linear regression offers the advantage of requiring only a very restricted quantity of log information. Second, linear regression is robust towards some of the forms of noise. In this paper, noise that may affect the approach result is considered to come in two different ways. It can appear as missing message occurrences or erroneous timestamps. Both cases were considered during experimentations.

In statistics, linear regression refers to any approach that models in a linear fashion the relationship between one or more variables denoted by y and one or more variables denoted by X. In such case, the denomination "linear model" is employed. When considering a data set $\{y_i, x_{i1}, ..., x_{ik}\}$, $i = 1, ..., n$ of statistical variables, a linear regression method can be applied between the dependent variable y_i and the vector of regressors x_i of size k *iff* the relationship between them is, at least approximately, linear. This linear relationship between y and x_i is expressed as $y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i = x_i'\beta + \epsilon_i$, $i = 1, ..., n$. For a more detailed introduction in the regression tools used in this paper please refer to [15]. In this paper, the linear regression inference is obtained by using the linear least squares approach.

Linear least squares (LLS) [6], [11] is a method employed for computing the linear regression and for fitting data to a mathematical or statistical model. The general problem can be stated as follows. Consider an overdetermined system:

$$\sum_{j=1}^{n} X_{ij}\beta_j = y_i (i = 1, ..., m) \quad (1)$$

of m simultaneous linear equations (SLE) in n unknown coefficients, β_1, $\beta_2, ...$, β_n with $m > n$, written in matrix form as $X\beta = y$ where:

$$X = \begin{pmatrix} X_{1,1} & \cdots & X_{1,n} \\ X_{2,1} & \cdots & X_{2,n} \\ \vdots & \ddots & \vdots \\ X_{m,1} & \cdots & X_{m,n} \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

In statistics, LLS is the computational foundation for ordinary least squares analysis (OLS), which is the method of regression analysis employed in this paper. The formulas for linear least squares fitting were derived by Gauss [13]. The algorithms in this paper employ LLS because LLS is at the same time simple and efficient in inferring the linear equations which describe the flow of messages in a business protocol. Ordinary least squares is the simplest and the most widely used method for estimating the parameter β. OLS is often employed to model experimental and observational data. Suppose b is a candidate value for an estimate of parameter β. Then the expression $y_i - x_i'b$ will be called the residual of i-th observation. The value of b which provides the minimal value for this expression will be called the *least squares estimator* for β.

TABLE I
OCCURRENCE LOG LINE OL_1 DERIVED FROM THE LOG IN TABLE II-B

LR	CT	MO	TT	LP
2	2	1	1	1

III. CORRELATION AND DISCOVERY APPROACH

In this section we provide some introductory examples and describe how the the process of correlating messages and inferring the business protocol is achieved. [1]

Example 1 Table I depicts the occurrence log OL_1 which records the number of occurrences for each type of message encountered in the log ML_1. An example of the logs considered in this paper is given in Table II. We recall that a message type is represented by its label. Each single row in OL_1 is deduced from an entire raw message log ML_i. Thus, an occurrence log can be seen as a very compact form of raw web service logs. This condensate form is at the basis of the approach presented here for achieving the correlation of messages.

A. Modeling the dynamics of BP messages.

Proposition 1 *The algebraic notation of a protocol is equivalent to its FSM representation in terms of descriptive power.*

Consider the protocol sample shown in Figure 2. In all the protocol figures shown in this paper, the dashed circles represent terminal states, as opposed to full circles that designate normal states. This example will describe how a set of linear equations describes the dynamics of messages in a business protocol with loops. The SLE provided in Equation 2 describes this protocol in algebraic terms. The equations marked with the same number of asterisks (*) or (**) are algebraically equivalent.

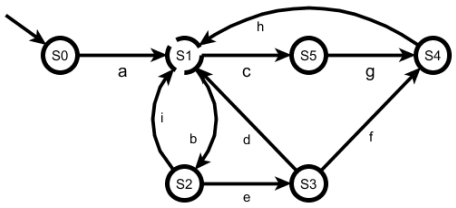

Fig. 2. Simplified business protocol.

[1]The theoretical proofs of the proposition and theorems, as well as additional results can be found at: *http://liris.cnrs.fr/kreshnik.musaraj/research/papers/ICWS-Tech-Report.pdf*

TABLE II
EXAMPLE OF RAW LOG ML_1 OF SOAP-BASED SERVICE EXECUTION MESSAGES

MessageID	Message label	SenderID	ReceiverID	Polarity	Message content	Timestamp
M_1	$LoginRequest$	192.168.5.1	192.168.15.4	$+$	$<?xml\ version = "1.0"\ ... >$	$2010:01:07\ 13:52$
M_2	$CommitTransaction$	192.168.15.4	192.168.5.1	$-$	$<?xml\ version = "1.0"\ ... >$	$2010:01:07\ 13:55$
M_3	$ModifyOperation$	192.168.5.1	192.168.15.4	$+$	$<?xml\ version = "1.0"\ ... >$	$2010:01:07\ 13:60$
M_3	$LoginRequest$	192.168.5.1	192.168.21.7	$+$	$<?xml\ version = "1.0"\ ... >$	$2010:05:07\ 07:15$
M_4	$CommitTransaction$	192.168.21.7	192.168.5.1	$-$	$<?xml\ version = "1.0"\ ... >$	$2010:05:07\ 07:26$
M_5	$TerminateTransaction$	192.168.5.1	192.168.21.7	$+$	$<?xml\ version = "1.0"\ ... >$	$2010:05:07\ 07:35$
M_6	$LogoutProcedure$	192.168.21.7	192.168.5.1	$-$	$<?xml\ version = "1.0"\ ... >$	$2010:05:07\ 07:46$

$$\begin{cases} 1.\ \bar{c} = \bar{a} - \bar{b} + \bar{d} + \bar{h} \\ 2.\ \bar{b} = \bar{a} + \bar{d} - \bar{c} + \bar{h} \\ 3.\ \bar{d} = \bar{e} - \bar{f}\ (*) \\ 4.\ \bar{e} = \bar{b} - \bar{i}\ (**) \\ 5.\ \bar{f} = \bar{e} - \bar{d}\ (*) \\ 6.\ \bar{g} = \bar{c} \\ 7.\ \bar{h} = \bar{f} + \bar{g} \\ 8.\ \bar{i} = \bar{b} - \bar{e}\ (**) \end{cases} \quad (2)$$

The method for constructing the SLE starts by considering each state at a time. Each linear equation is related to a state and provides the flow of messages in and out that particular state. Since there are several potential messages entering or leaving a state, then there may be several equations corresponding to a state. An equation describes a linear relationship between the number of occurrences of all messages that are directly related to a particular state. A given variable (a label with a bar) in an equation represents the number of occurrences of the message having the same label as the variable. On the right side of the equality sign are located all the other variables related to the remaining messages of that same state. Each variable is associated with a coefficient. The variables on the right side have either a $+$ or $-$ sign. This sign is very important since $+$-signed variables represent messages that exit the corresponding state, and negative-signed variables stand for messages entering that state. Theoretically, all coefficients are ± 1, since a message can neither be split in two, nor created or disappear. Nevertheless, since the logs we consider here may exhibit defaults, such as missing messages, and because of existing loops then the experimental coefficients may be non-integer values. Experimental results however allow to define the interval inside which a coefficient value is considered as trustworthy. Moreover, we exploit the high-value coefficients to detect the existence of complex structures with unprecedented ease.

Consider the generic form of the state of a business protocol depicted in Figure 3.

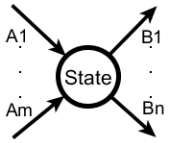

Fig. 3. General form of a protocol state.

For a given existing message A_k, $1 \leq k \leq m$ the following equation can be stated:

$$\bar{A_k} = \sum_{j=1}^{n} \bar{B_j} - \sum_{l=1}^{m} \bar{A_l} + \bar{A_k} = \sum_{j=1}^{n} \bar{B_j} - \sum_{l=1, l \neq k}^{m} \bar{A_l} \quad (3)$$

where $\bar{A_k}$ denotes the number of occurrences of message type A_k that have transited outside state $State$. Note the difference between the number of types of messages (denoted by n) and the number of occurrences of each type of message. A type of message example corresponds simply to a message label e.g. *"browseProducts"*. Equation (3) is correct since the total of messages occurrences entering a state is obviously the same as the number of those exiting the same state. In other words:

$$\sum_{j=1}^{n} \bar{B_j} - \sum_{l=1}^{m} \bar{A_l} = 0 \quad (4)$$

One can now see that Equation (3) is obtained by adding $\bar{A_k}$ to both sides of Equation (4) which is arithmetically equivalent. Moreover, Equation (4) states the law of conservation of the number of message occurrences.

B. Algorithmic procedures

Let us introduce the algorithms that exploit the SLE approach in different ways.

1) The naive approach: the $n-delta$ algorithm: The naive version of the algorithm uses as input the log containing all message occurrences recorded during the web service activity. The corresponding pseudo-code is shown in Algorithm 1. The algorithm proceeds by taking a message type vector at a time, and by computing the coefficients of all the possible equations that can be obtained throughout all possible combinations of message type in equations of size i. The provided result is the set of correlation matrixes. The logical AND-sum of all the correlation matrix yields the final correlation matrix.

Theorem 1 *The complexity of the naive-delta (n-delta) algorithm is $O(2^n)$*

The optimal version of the $delta$ algorithm, which is shown in Algorithm 2, uses the same input as $naive - delta$. The algorithm proceeds in a much simpler fashion by taking a message type vector at a time, and by computing *only* the coefficients of the equations of size $n - 1$. This algorithm provides directly the final correlation matrix.

Algorithm 1 $naive - delta$

Require: Occurrence log A of n message types
Ensure: M_{xi} the set of correlation matrixes // $n-1$ *in total*
1: **for** $i = 1$ to $n-1$ **do**
2: $B = A$
3: **while** $B \neq \emptyset$ **do**
4: Choose an occurrence vector $b \in B$
5: Remove the column of b from B
6: $Rs = B$ // *Rs is the matrix of regressors*
7: $RegressorsSet = combinations(Rs, i)$ // *the set of non-redundant combinations of i elements.*
8: **while** $RegressorsSet \neq \emptyset$ **do**
9: Choose $r_i \in RegressorsSet$
10: // r_i *is an occurrence vector of size* i
11: Remove the column of r_i from $RegressorsSet$
12: $C = OrdinaryLeastSquares(r_i, b)$
13: Insert correlation vector C in matrix M_{xi}
14: **end while**
15: **end while**
16: **end for**

Algorithm 2 $delta$

Require: Occurrence log A of n message types
Ensure: M_x the correlation matrix
1: $B = A$
2: **while** $B \neq \emptyset$ **do**
3: Choose an occurrence vector $b \in B$
4: Remove the column of b from B
5: $Rs = A$ // *Rs is the matrix of regressors*
6: Remove the column of b from Rs
7: $C = OrdinaryLeastSquares(Rs, b)$
8: Insert correlation vector C in matrix M_x
9: **end while**

2) The optimal approach: delta algorithm: The complexity of Algorithm 2 is sensibly lower, as shows the following theorem.

Theorem 2 *The complexity of the delta algorithm is* $O(n^2)$

In most web service protocols the number of message type n is generally low. To give an idea on the order of magnitude, very large protocols such as eBay Trading service achieve a maximum of 64 message types [10]. Thus the square polynomial complexity has no significant impact on the overall performance of the algorithm. On the other hand, what is of a much greater influence is the number of message occurrences stored into logs, since their number can reach very large values. The complexity of the algorithm that counts the number of occurrences for each message type will then determine the overall complexity of the approach. If N is the number of all the different message occurrences stored in the service logs and M the number of message types, then the complexity of the counting algorithm is $O(M \times N)$. Since N is by many orders of magnitude greater than M, thus the resulting complexity is $O(N)$. In conclusion, the occurrence counting algorithm has a linear complexity. A direct consequence of this result is that the overall process of extracting the SLE that allows to mine the service business

TABLE III
GENERAL FORM OF THE CORRELATION MATRIX WITH METHOD RESULT.

	a_1	a_2	.....	a_{m-1}	a_m
a_1	0	$i_{1,2}$	.....	$i_{1,m-1}$	$i_{1,m}$
a_2	$i_{2,1}$	0	.....	$i_{2,m-1}$	$i_{2,m}$
...	...	...	.....	...	...
...	...	...	.....	...	...
...	...	...	.....	...	...
a_{m-1}	$i_{m-1,1}$	$i_{m-1,2}$	.....	0	$i_{m-1,m}$
a_m	$i_{m,1}$	$i_{m,2}$	.....	$i_{m,m-1}$	0

TABLE IV
SINGLE OCCURRENCE VECTOR

	a
a	0
b	+1
c	−1
d	+1
e	0
f	0

protocol has an remarkably good performance. The minor influence of the polynomial sub-algorithm was confirmed by experimental results on synthetic logs, as is shown in Section IV.

C. Result interpretation and visualization

The result provided by the algorithm is a matrix composed of correlation vectors. The vectors correspond to the columns of the matrix. This correlation matrix allows to easily obtain the SLE corresponding to a protocol. Table III shows the generic form of this array. Let m be the number of message types in a protocol and a_1, a_2, ... a_m, the message types that need to be correlated. The correlation matrix provides the coefficients that are to be used in an equation involving a variable a_i on the left side, and variables $a_j, (j \neq i)$ on the right side. If we have the matrix column shown in Table IV, the corresponding equation that is derived from this column is:

$$1 \times \bar{a} = 1 \times \bar{b} - 1 \times \bar{c} + 1 \times \bar{d} + 0 \times \bar{e} + 0 \times \bar{f} = \bar{b} - \bar{c} + \bar{d} \quad (5)$$

This equation describes two possible cases, that are illustrated in Figure 4. Note that the only difference between the two cases is the direction of message flow. In Figure 4(a), messages b, d enter the state and a, c exit it, while in (b) we have the reverse situation. The choice between these two possibilities is made by employing the timestamps of message occurrences to establish order relationships.

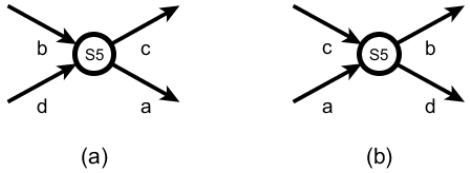

Fig. 4. Graph-represented equivalent of a linear equation where incoming messages (a) become outgoing (b).

TABLE V
OCCURRENCE LOG OF SAMPLE PROTOCOL IN FIGURE 5

a	b	c	d	e
10	5	5	3	2
12	7	5	3	4
15	8	7	5	3
20	13	7	9	4
31	14	17	8	6

TABLE VI
COEFFICIENT MATRIX COMPUTED FROM TABLE V

	a	b	c	d	e
a	0	0	1	0	0
b	1	0	0	1	1
c	1	0	0	0	0
d	0	1	-1	0	-1
e	0	1	-1	-1	0

The matrix is to be interpreted as follows: we consider each column at a time. If for row k and column l the value i_{kl} is 0, then the message type corresponding to that row is not correlated to the message type associated with column l. Thus, each column expresses via a proper linear equation the correlation between the corresponding message type and the other message types. Note in Table III that the diagonal values of the correlation matrix are always zero. This is done arbitrarily since the correlation between a message type and itself is always valid, thus not relevant. Furthermore, experiments have shown that for a correlation to be correctly estimated, the coefficient values a_i are to be found in the interval: $0.91 < a_i < 1.09$.

IV. CORRELATION AND DISCOVERY: A USE-CASE EXAMPLE

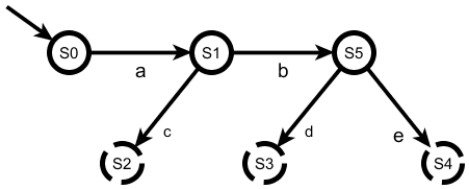

Fig. 5. Use-case for correlation and business protocol discovery.

Let us illustrate how the approach achieves the correlation of messages and protocol discovery using the following example.

Example 2 Consider the simple protocol in Figure 5. Table V shows the occurrence log deduced from message logs issued by the protocol in Figure 5. When the algorithm is run on this log, we obtain the resulting matrix in Table VI. We observe that the result corresponds to the expected solution. Moreover we observe that the condition $i < j$ is necessary because it provides the most convergent solution that the LSF method can find on a rank deficient occurrence log.

Starting from the correlation matrix in Table VI we obtain the linear system shown in Equation 6. This linear system leads to the straightforward states shown in Figure 6. Equation 1 is translated into the first state (Figure 6(a)), while equations 2, 4, 5 provide the state shown in Figure 6(b) and equation 3 gives the state (c) of the same figure. From the SLE in Equation 6 we see that $\bar{b} = \bar{d} + \bar{e}$ and the log timestamps of b always precede those of d and e. Thus, the only possible outcome that unifies these three states is the one already shown in Figure 5.

Loops are easily identified by the fact that the coefficients of message types generated by loops have an absolute value much greater than 1. For more details on the reasons behind, please refer to the technical report (link provided as a footnote, Section III).

$$\begin{cases} 1.\ \bar{a} = \bar{b} + \bar{c} \\ 2.\ \bar{b} = \bar{d} + \bar{e} \\ 3.\ \bar{c} = \bar{a} - \bar{d} - \bar{e} \\ 4.\ \bar{d} = \bar{b} - \bar{e} \\ 5.\ \bar{e} = \bar{b} - \bar{d} \end{cases} \quad (6)$$

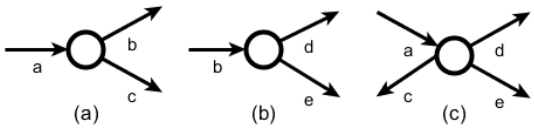

Fig. 6. States obtained from (a) equation 1, (b) equations 2, 4, 5, and (c) equation 3 from the linear system in Equation 6.

TABLE VII
IMPACT OF M ON SCALABILITY AND PERFORMANCE.

# Msg. type (M)	Min. MT (s.)	Max. MT (s.)	Avg. MT(s.)
10	0.004	0.030	0.017
25	0.010	0.060	0.035
50	0.070	0.140	0.105
75	0.190	0.280	0.235
100	0.370	0.520	0.445
200	3.170	4.960	4.065

V. EXPERIMENTAL RESULTS

Synthetic data generated by a business protocol simulator were used to test the correctness and performance of the algorithms.[2] Experiments were conducted to study the scalability of $delta$. The influence of noise was also investigated by introducing errors in the logs. The desired percentage of errors introduced was variable so that the evolution of the behavior of the algorithm would be clearer. The business protocols employed for synthetic data generation are of realistic sizes (up to 100 message types and 10^6 web service instance conversations). The implementation of the algorithm and the experiments are conducted using Matlab. Nevertheless, the migration towards other mathematical software requires little effort.

Table VII shows the experimental results on the comparative performance of the $delta$ algorithm versus the scalability of

[2]The simulation tool can be downloaded at *http://liris.cnrs.fr/kreshnik.musaraj/technology/simulation/index.html*. The Matlab models used as well as the source code of the algorithms can be downloaded at *http://liris.cnrs.fr/kreshnik.musaraj/technology/ws/index.html*

TABLE VIII
IMPACT OF N ON SCALABILITY AND PERFORMANCE.

# Msg. occ. (N)	Min. MT (s.)	Max. MT (s.)	Avg. MT(s.)
1000	0.10	0.50	0.30
2000	0.30	1.10	0.65
5000	0.92	2.50	1.71
10000	2.10	3.62	5.13
50000	27.50	32.06	29.78
1000000	57.01	67.29	62.15

TABLE IX
IMPACT OF NOISE ON COEFFICIENT ESTIMATION WITH OLS

Δ #Occurrences (%)	Avg. Δ #Coefficients (%)
1	$\pm$ 0.03
5	$\pm$ 0.7
10	$\pm$ 15.7
20	$\pm$ 45.0

message types. $Min.\ MT$, $Max.\ MT$ and $Avg.\ MT$ stand for respectively Minimal, Maximal and Average Measured Time. Table VII clearly proves that employing OLS for constructing the correlation matrix is extremely efficient. Table VIII depicts instead the same comparative performance of the algorithm but this time it is based on the number of message occurrences. One should observe the convergence of $Min.\ MT$ and $Max.\ MT$ towards $Avg.\ MT$ for increasing values of M and N.

Table IX shows the impact of noise on the values of coefficients estimated using OLS. The first column provides the difference in percentage between the accurate number of occurrences and the value reported on the noisy occurrence log. The second column shows the percentage of divergence between the coefficients estimated using perfect data and the values estimated using message logs subject to missing message occurrences. It is important to notice that, when exposed to incorrect occurrence logs, the method will provide the correlation matrix that best fits the data. In this sense, the resulting protocol will be adapted to each occurrence log, thus the result will evolve as a function of the accuracy of the log. Nevertheless, noise impacts mainly the data contained in the recorded messages, and the rate of missing messages is much lower than the rate of incorrect data. The only exception is the case of error-prone log software, which goes beyond the scope of this paper.

VI. RELATED WORK

The business protocol discovery problem addressed in [19] and [16] can be considered as a particular case of a more general issue: the extraction of a model from its instances. Literature related to model discovery is extensive, for example in grammatical inference [2], [20], in workflow mining [1], [7], [14], [23], or in Web services interaction mining [9]. In grammatical inference, the problem consists in finding a grammar generating a language, given a set of words that belong to this language, and a set of words that do not belong to it; using both positive and negative examples allows producing a correct model. The business protocol discovery problem is different in the sense that only (noisy and incomplete) positive instances are available. In Web services interaction mining, the goal is to discover from logs a workflow modelling the interactions that take place between several services. Despite the similar context, this differs from business protocol discovery in its knowledge extraction level; in [16], [19] cross-services protocols are not considered. Workflow mining (or software process discovery) is very similar to business protocol discovery. The main differences between all techniques lie in the choice of the model (automaton [7], [16], [19], Petri net [23] or directed graph [1], [14]), in the fact that noise is considered [1], [7], [14], [16], [19] or not [23], and that the extraction process allows user driven refinement [16], [19] or not [1], [7], [14], [23]. Recent improvements in the discovery of process models are provided in [12] where the authors address the problem of discovering the process model when the event log is provided as an unlabelled stream of events; and [5] where we encounter a set of techniques for employing the theory of regions to transform a log into a Petri net.

At the same time the work reported in [16], [19] was achieved, similar techniques [8] were developed, though with quite different concerns. In [8] the concepts of *interaction* and *interaction protocol* are used equivalently to *conversation* and *business protocol*. The goal is to solve an interoperability problem between a client and several Web services that expose the same interface but may have different interaction protocols. The proposed solution consists in automatically extracting approximated interaction protocols from recorded interactions (between these services and previous users). The main advantage of the approach presented in [8] is that it is fully automatic, which is essential in a service discovery architecture. However, this can be an important drawback in the service management and re-engineering area, where it is essential to allow a user driven refinement of the extracted model [16], [19], [21].

It is clear from this bibliography analysis that in the Web services field, the majority of works focus on the process discovery (workflows) and protocol discovery [19], [21], [22]. The scope of this paper is limited to the issue of protocol discovery. There are two main internal directions in this field: protocol discovery using conversation logs (messages log with conversation identifiers) and using message logs (messages log without conversation identifiers). The first direction has been investigated extensively. Meanwhile, the issue of dealing with message logs without conversation identifiers is a topic that has started to gain importance only during recent years [21], [22]. In [22] the authors model logs using graphs and employ graph theory techniques to extract the conversations and build the Business Protocol. For converting the initial message graph into a reduced one, the authors use the Dempster-Shafers mathematical theory of evidence. While in [21], two types of correlation information are used for conversation discovery. The first one is key-based correlation, in which a unique identification value is used for correlating messages in a single conversation. The other type is reference-based (or also known as chain-based), whose principle is using a reference which

points to the preceding message in the conversation sequence.

Despite their importance, the existing approaches either require strong assumptions such as the conversation identifier [23], sufficient information in logs [21], or hypothesis that might not be correct in realistic scenarios [22]. This urges for an approach that requires as few assumptions as possible while presenting a high performance, offering an optimal accuracy and robustness towards noise.

VII. Conclusions and future work

Conversation mining by means of message correlation for web services business protocol mining is an important step in SOA architectures. This domain requires further attention since it is far from being exhaustively explored. This motivates the efforts on establishing automatic methods for message correlation relying on virtually no assumptions on the logs used as starting point. This concerns not only the properties of log data (absence of noise, statistical properties etc.) but also the data content itself, which is subject to change in large measures depending on the context and SOA implementations. Our contributions are: (i) Correlation of messages using almost no information aside message timestamps, (ii) Modeling of the the dynamic flow of messages into a business protocol without any prior knowledge, (iii) Business protocol generation from the algebraic description of the flow of messages, and (iv) A linear-complexity algorithm for the correlation and discovery process. We provided an algebraic form that is equivalent to the finite-state machine notation of a business protocol to continue on obtaining the linear system that described a business protocol by means of linear regression techniques. This was achieved via the Ordinary Least Squares estimator that was shown to be noise-resistent while providing the coefficients of the linear equations. The estimator allowed to obtain a low-complexity / high performance algorithm that is capable of dealing with huge logs.

In the future we intend to combine the $delta$ algorithm with other methods that already provide protocol mining from conversation logs. An on-going effort is addressing the extension of the algorithm into the Business Process mining domain. In order to achieve this, the $delta$ algorithm needs to be adapted to account for process-specific patterns such as parallel transitions, AND-splits, AND-joins etc. Preliminary results are very optimistic and show that this approach has generic capabilities that go beyond technical details of service logs and domain-related particularities. Another perspective that we plan to investigate is the adaptation of the $delta$ algorithm in order to extract cross-organizational service protocol, and to mine logs generated from composite web services.

Acknowledgment

This work is partially funded by the EU Framework 7 STREP project COMPAS (215175, FP7-ICT-2007-1).

References

[1] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *EDBT '98*, pages 469–483, Valencia, Spain, Mar 1998. Springer.

[2] D. Angluin and C. H. Smith. Inductive inference: Theory and methods. *ACM Computing Surveys*, 15(3):237–269, Sep 1983.

[3] B. Benatallah, F. Casati, J. Ponge, and F. Toumani. On temporal abstractions of web services protocols. In *CAiSE '05 Short Paper Proceedings*, pages 39–44, Porto, Portugal, June 2005. Springer.

[4] B. Benatallah, F. Casati, and F. Toumani. Analysis and management of web service protocols. In *Conceptual Modeling - ER '04*, pages 524–541, Shanghai, China, Nov 2004. Springer.

[5] J. Carmona, J. Cortadella, and M. Kishinevsky. Divide-and-conquer strategies for process mining. In *BPM '09: Proceedings of the 7th International Conference on Business Process Management*, pages 327–343, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] S. Chatterjee and A. S. Hadi. *Regression Analysis by Example, 3rd ed.*, chapter 2, pages 21–50. Wiley-Interscience, New York, 2000.

[7] J. E. Cook and A. L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, July 1998.

[8] G. Denaro, M. Pezzé, D. Tosi, and D. Schilling. Towards self-adaptive service-oriented architectures. In *TAV-WEB '06*, pages 10–16, Portland, Maine, USA, Jul 2006. ACM.

[9] S. Dustdar, R. Gombotz, and K. Bana. Web services interaction mining. Technical Report TUV-1841-2004-16, Technical University of Vienna, Vienna, Austria, Sep 2004.

[10] eBay Developers Program. The ebay trading api, available at *http://developer.ebay.com/products/trading/*, 2010.

[11] A. L. Edwards. *Introduction to Linear Regression and Correlation*, chapter 3, pages 20–32. W.H.Freeman & Co Ltd, San Francisco, 1976.

[12] D. R. Ferreira and D. Gillblad. Discovering process models from unlabelled event logs. In *BPM '09: Proceedings of the 7th International Conference on Business Process Management*, pages 143–158, Berlin, Heidelberg, 2009. Springer-Verlag.

[13] C. F. Gauss. *Theoria combinationis obsevationum erroribus minimis obnoxiae, Werke Vol. 4*. Gottingen, Germany, 1823.

[14] S.-Y. Hwang and W.-S. Yang. On the discovery of process models from their instances. *Decision Support Systems*, 34(1):41–57, Dec 2002.

[15] I. C. F. Ipsen. *Numerical Matrix Analysis: Linear Systems and Least Squares*. Society for Industrial and Applied Mathematics, January 2009.

[16] H. R. Motahari-Nezhad, R. Saint-Paul, B. Benatallah, F. Casati, J. Ponge, and F. Toumani. Servicemosaic: Interactive analysis and manipulations of service conversations. In *ICDE '07*, Istanbul, Turkey, Apr 2007. IEEE. Demonstration.

[17] L. Măruşter, A. J. Weijters, W. M. Aalst, and A. Bosch. A rule-based approach for process discovery: Dealing with noise and imbalance in process logs. *Data Mining in Knowledge Discovery*, 13(1):67–87, 2006.

[18] H. R. M. Nezhad, B. Benatallah, R. Saint-Paul, F. Casati, and P. Andritsos. Process spaceship: discovering and exploring process views from event logs in data spaces. *VLDB*, 1(2):1412–1415, 2008.

[19] H. R. M. Nezhad, R. Saint-Paul, B. Benatallah, and F. Casati. Protocol discovery from imperfect service interaction logs. In *ICDE '07*, pages 1405–1409, Istanbul, Turkey, Apr 2007. IEEE.

[20] R. Parekh and V. Honavar. Grammar inference, automata induction, and language acquisition. In *Handbook of natural language processing*, pages 727–764. Marcel Dekker, Inc., New York, USA, 2000.

[21] M. N. H. Reza. *Discovery and adaptation of process views*. PhD thesis, Computer Science and Engineering, Faculty of Engineering, UNSW, Sydney, Australia, 2008.

[22] B. Serrour, D. P. Gasparotto, H. Kheddouci, and B. Benatallah. Message correlation and business protocol discovery in service interaction logs. In *CAiSE*, pages 405–419, 2008.

[23] W. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, Sep 2004.

[24] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst. The prom framework: A new era in process mining tool support. In *ICATPN*, pages 444–454, 2005.