Improving Event Correlation for Non-process Aware Information Systems

Ricardo Pérez-Castillo[1], Barbara Weber[2], Ignacio García-Rodríguez[1] and Mario Piattini[1]
[1]*Instituto de Tecnologías y Sistemas de Información (ITSI), University of Castilla-La Mancha, Paseo de la Universidad 4, 13071, Ciudad Real, Spain*
[2]*University of Innsbruck, Technikerstraße 21a, 6020, Innsbruck, Austria*
{ricardo.pdelcastillo, ignacio.grodriguez, mario.piattini}@uclm.es, barbara.weber@uibk.ac.at

Keywords: Enterprise Modelling, Business Process, Event Correlation, Legacy Information Systems.

Abstract: Business process mining is a solution to discover business processes. These techniques take event logs recorded by process-aware information systems. Unfortunately, there are many traditional systems without mechanisms for events collection. Techniques for collecting events (which represent the execution of business activities) from non-process-aware systems were proposed to enable the application of process mining to traditional systems. Since business processes supported by traditional systems are implicit, correlating events into their execution instances constitutes a challenge. This paper adapts a previous correlation algorithm and incorporates it into a technique for obtaining event logs from traditional systems. This technique instruments source code to collect events with some additional information. The algorithm is applied to the events dataset to discover the best correlation conditions. Event logs are built using such conditions. The technique is validated with case study, which demonstrates its suitability to discover the correlation set and obtain well-formed event logs.

1 INTRODUCTION

Current companies must continuously evolve to maintain their competitiveness levels. Keeping this in mind, process modelling is essential for companies to be able to understand, manage and adapt their business processes (Weske, 2007). Despite this important fact, a vast amount of companies do not model their business processes. When these companies decide in favor of business process modelling, they have two main options: (i) modelling from scratch by business experts, which is time-consuming and error-prone; (ii) using business process mining techniques to discover business processes from system execution information (van der Aalst and Weijters, 2005). The focus of this paper is on the second option since it takes into account the business knowledge embedded in enterprise information systems.

Business process mining techniques allow for extracting information from process execution logs – known as event logs (van der Aalst and Weijters, 2005). Event logs contain information about the start and completion of activities and the resources executed by the processes (Castellanos, Medeiros et al., 2009). These events logs are often recorded by process-aware information systems (PAIS) (e.g., Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) systems). The process-aware nature of PAIS facilitates direct events recording during process execution. However, not all information systems are process-aware. In fact, there is a vast amount of enterprise information systems which are non-process-aware (termed as traditional systems in this paper) though they could also benefit from the application of process mining techniques.

Previous works made a particular effort for the registration of event logs from traditional systems. Firstly, main challenges involved in the collection of event logs from traditional systems were identified (Pérez-Castillo et al., 2010). Secondly, a particular technique to obtain event logs from traditional systems was also developed (Pérez-Castillo et al., 2011). This technique first injects statements into the source code to instrument it. The instrumented system is then able to record some events, which are finally analysed and organized in an event log. This technique has already been applied to real-life traditional systems with promising results (Pérez-

Castillo et al., 2011); (Pérez-Castillo et al., 2011). However, the accuracy of the previous technique is limited, since events were correlated in different process instances by using some simple heuristics. In fact, event correlation is a key challenge, i.e., each event must be assigned to the correct instance, since it is possible to have various instances of the business process running at the same time. This paper therefore addresses the weaknesses on the previous technique by providing an enhanced technique, which adapts and applies an existing event correlation algorithm (Motahari-Nezhad et al., 2011).

The new technique relies on human interaction to firstly identify some candidate correlation attributes. The candidate attribute values are then recorded with each event by means of an instrumented version from a traditional system. After that, event correlation algorithms proposed in a related work (Motahari-Nezhad et al., 2011) are adapted and applied to this intermediate information to discover the sub-set of correlation attributes and conditions. The correlation set is finally used to obtain accurate process instances in the final event log. For a smoother introduction into industry, this technique has been developed by using database-stored intermediate information as well as a set of algorithms implemented as stored procedures. Since the technique is tool-supported, a case study involving a real-life traditional system has been conducted to demonstrate the feasibility of the proposal. The empirical validation results show the technique is able to obtain the set of correlation attributes allowing appropriate event logs in a moderate time.

The paper is organized as follows: Section 2 summarizes related work; Section 3 presents in detail the proposed technique; Section 4 conducts a case study with an author management system; and Section 5 discusses conclusions and future work.

2 RELATED WORK

Event correlation is an issue of growing importance in the process mining field due to the increasing heterogeneity and distribution of enterprise information systems. In addition, there are various ways in which process events could be correlated. In fact, many times event correlation is subjective and most proposals employ correlation heuristics (McGarry, 2005). Most techniques assess some indicators and check if they are under or above a heuristic threshold to discard non-promising correlation attributes. For example, (Burattin and Vigo, 2011) propose an approach consisting of the introduction of a set of extra fields, decorating each single activity log. These attributes are used to carry the information on the process instance. Algorithms are designed using relation algebra notions, to extract the most promising case IDs from the extra fields. Other techniques proposals, as in (Rozsnyai et al., 2011), are based in the use of algorithms to discover correlation rules by using assessments of statistic indicators (e.g., variance of attribute values) from datasets. Similarly, (Ferreira and Gillblad, 2009) propose a probabilistic approach to find the case ID in unlabeled event logs.

(Motahari-Nezhad et al., 2011) propose a set of algorithms to discover the most appropriate correlation attributes and conditions (e.g., conjunctive and disjunctive conditions grouping two or more correlation attributes) from the available attributes of web services interaction logs.

This paper presents an improvement of a previous technique to retrieving event logs from traditional systems (Pérez-Castillo et al., 2011), by adapting and applying the algorithm to discover the correlation set provided by (Motahari-Nezhad et al., 2011). While this algorithm is applied to web services logs, the current approach adapts the algorithm to be applied in traditional information systems for obtaining event logs. The feasibility of this approach is empirically validated by means of a case study.

Moreover, there are proposals addressing the distribution of heterogeneous event logs. For example, (Hammoud, 2009) presents a decentralized event correlation architecture. In addition, *Myers et al.* (Myers et al., 2010) apply generic distributed techniques in conjunction with existing log monitoring methodologies to get additional insights about event correlation. Decentralized event correlation approaches remain however outside of the scope of this paper.

3 EVENT CORRELATION

Event correlation deals with the definition of relationships between two or more events so to point out events belonging to a same business process execution (i.e., process instance). Event correlation is very important in traditional information systems, since the definitions of the executed business processes are not explicitly identified (Pérez-Castillo et al., 2010). Figure 1 shows an overview of the event correlation challenge. Each business process

can be executed several times. Each execution is known as process instance. Events collected during system execution must be correlated into the correct instance.

This paper presents a technique to obtain event logs from traditional systems, paying special attention to the event correlation improvement regarding previous work. The technique consists of three stages. Firstly, the technique records events from the execution of traditional systems. During this stage the technique allows experts to identify candidate correlation attributes, whose runtime values will then be collected together with each event. As a result, events and their respective attributes are then stored in a database in an intermediate format (cf. Section 3.1). Secondly, the algorithm proposed by (Motahari-Nezhad et al., 2011) is adapted and applied with the event datasets so to discover the most appropriate set of attributes and conditions for the events correlation (cf. Section 3.2). Finally, the third stage applies an algorithm based on the correlation set in order to correlate each event with its corresponding process instance (cf. Section 3.3). As a result, a standard-format event log is obtained from the source traditional system.

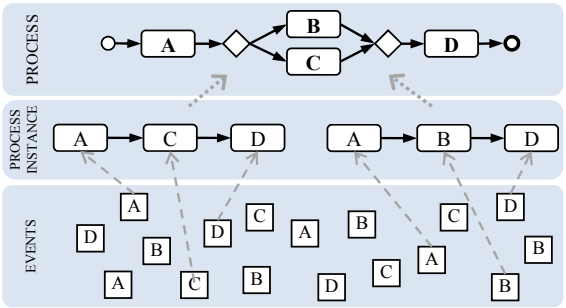

Figure 1: Event correlation overview.

3.1 Event Collection

The event collection stage is in charge of the suitable generation and storage of events throughout system execution. Since traditional information systems do not have any in-built mechanism to record events about executed business processes, this stage instruments information systems to record events. The instrumentation is semi-automated by a parser that syntactically analyzes the source code and injects statements in particular places of the code to record events during system execution.

This work follows the *'a callable unit / a business activity'* approach proposed by *Zou et al.* (Zou and Hung, 2006). Callable units are the generic elements (e.g., Java methods, C or COBOL procedures, etc.) in which the parser injects statements to record an event corresponding to the execution of a business activity. Despite this fact, not all the executions of callable units have to be recorded as events. Some callable units such as fine-grained or technical callable units do not correspond to events and must be discarded. The injection in the exact place is consequently supported by some information provided by experts. Such experts (i) delimit business processes with the start and end callable units of each process; (ii) establish the boundaries of non-technical source code to be instrumented; and finally, (iii) they identify those code elements that can be treated as candidate correlation attributes.

This stage is supported by an improved version of the tool presented in (Pérez-Castillo et al., 2011), which supports the identification and addition of candidate correlation attributes. Selection of candidate correlation attributes is a key task, since an incomplete list of candidate attributes may lead to a non-suitable correlation. This stage provides experts with all the possible selectable attributes. These attributes are every parameter appearing in callable units as well as the output and fine-grained callable units that are invoked within those callable units, which are considered to be collected as events. The information about candidate correlation attributes is injected together with a statement enabling event collection.

```
public class SocioFacadeDelegate {
    [...]
    public static void saveAuthor(AuthorVO author) throws InternalErrorException {
        writeDBEvent("SocioFacadeDelegate.saveAuthor", "Author Management", "", "start",
          false, false, -1, false, 2, 8, "", "" + author.getId(), "" + author.isHistorico(),
          "" + author.getNumeroSocio(), "", "" + author.getCotas());
        try {
            SaveAuthorAction action = new SaveAuthorAction(author);
            PlainActionProcessor.process(getPrivateDataSource(), action);
        } catch (InternalErrorException e) {
            throw e;
        } catch (Exception e) {
            throw new InternalErrorException(e);
        }
        writeDBEvent("SocioFacadeDelegate.saveAuthor", "Author Management", "", "complete",
          false, true, -1, false, 2, 8, "", "" + author.getId(), "" + author.isHistorico(),
          "" + author.getNumeroSocio(), "", "" + author.getCotas());
    }
    [...]
}
```

Figure 2: Example of code instrumentation.

Figure 2 provides an example of the results obtained after instrumenting a Java method of the system under study (cf. Section 4). The two *tracing* statements (see highlighted statements) are injected at the beginning and at the end of the body of the method. Those candidate correlation attributes that are present in a method (e.g., a parameter or variable) are automatically injected in the *tracing* statements, i.e., the respective variables are in the set of parameters of the invocation to the method *'writeDBEvent'* (see Figure 2). However, not all correlation attributes defined by experts are present in all methods (e.g., due to the absence of a particular variable). In this case, the respective

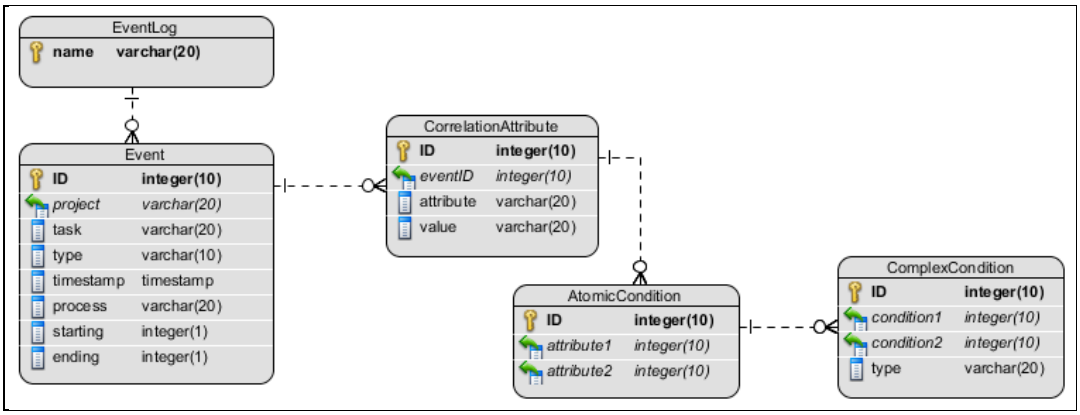

Figure 3: Database schema for event and attributes.

parameter of the method *'writeDBEvent'* (i.e., the tracing statement) is an empty string (" "). As a result, during execution of this method, the runtime value (or an empty value) will be recorded together with the name of the attribute and the event (the name of the method representing the business activity).

The instrumented system is then normally executed and -when an injected statement is reached- it records events together with the value of all the candidate correlation attributes available in that callable unit. Unlike other similar techniques, it does not build an event log on the fly during system execution (Pérez-Castillo et al., 2011); Pérez-Castillo et al., 2011). Instead, this technique stores all the information about events and their candidate correlation attributes in a database. A relational database context facilitates the implementation of faster algorithms to discover the correlation set from large datasets (cf. Section 3.2).

Figure 3 shows the database schema used to represent the intermediate event information. The *EventLogs* table is used to represent different logs obtained from different source systems. The *Events* table contains all the different events collected, including the executed task, type (start or complete), originator, execution timestamp, two columns to indicate if the executed task is the initial or final task of a process, as well as the process name. *CorrelationAttributes* is a table related to *Events* and contains the runtime values of candidate correlation attributes.

Candidate correlation attributes are combined by means of correlation conditions which are then used to correlate events. We differentiate two kinds of correlation conditions: atomic and complex conditions. Firstly, atomic conditions represent key-based conditions which compare two correlation attributes. For example, *condition1:attribute1 =attribute2* signifies that two events will be correlated if the value of *attribute1* of the first event is equal to the value of *attribute2* of the second event under evaluation. These conditions are stored in the *AtomicConditions* table (see Figure 3). Secondly, complex conditions evaluate two different conditions at the same time that are combined by a logic operator, e.g., conjunction (*AND*) or disjunction (*OR*) (Motahari-Nezhad et al., 2011). For example, *condition3: condition1 AND condition2* evaluated for two events signifies that both *condition1* and *condition2* must be met for the couple of events at the same time. Table *Complex-Conditions* represents this information in the database schema (see Figure 3).

3.2 Discovering Correlation Attributes

After collection of events and candidate correlation attributes, an adaptation of the algorithm described in (Motahari-Nezhad et al., 2011) is applied to discover the most appropriate correlation set. This set consists of a set of atomic conditions (i.e., equal comparisons between couples of attributes) as well as a set of complex conditions (i.e., conjunctive comparisons between couples of atomic conditions). The technique does not consider disjunctive conditions since these conditions are only needed for heterogeneous systems to detect synonyms of some correlation attributes.

The algorithm (see Figure 4) first considers all the possible combinations of attributes involved in atomic conditions (lines 1-3), and it then prunes the non-interesting conditions (i.e., less promising conditions) using the three following heuristics.

Heuristic 1. When attributes of atomic conditions are the same, the distinct ratio must be above alpha or distinct to one (line 4). *Distinct Ratio* (Eq. 1)

indicates the cardinality of an attribute in the dataset regarding its total number of non-null values. When it is under alpha or one it signifies that the attribute contains a global unique value and the atomic condition with this attribute can therefore be pruned. *Alpha* (Eq. 3) is the threshold to detect global unique values and it indicates how much values vary regarding the size of the dataset.

Heuristic 2. If the atomic condition is formed by two different attributes it is pruned when the shared ratio is above the alpha threshold (line 5). In a similar way the previous heuristic, *Shared Ratio* (Eq. 2) represents the number of distinct shared values regarding their non-null values for both attributes.

Input: *Attributes*; *Events*
Output: *AC*: the set of atomic conditions; *CC*: the set of conjunctive conditions
1: **for** a ∈ *Attributes* ∧ b ∈ *Attributes* **do**
2: *AC*← "a = b"
3: **end for**
4: *AC*←*AC* – { c | c.attribute1 = c.attribute2 **and** (*DistinctRatio* (c.attribute1) < α **or** *DistinctRatio* (c.attribute1) = 1) }
5: *AC*←*AC* – { c | c.attribute1 ≠ c.attribute2 **and** *SharedRatio* (c.attribute1, c.attribute2) <α }
6: *AC*←*AC* – { c | *PIRatio* (c.attribute1, c.attribute2) < α **or** *PIRatio* (c.attribute1, c.attribute2) >β }
7: N_0←*AC*; N_1← { }
8: k ← 1
9: **for** c1 ∈ N_{k-1} **and** c2 ∈ N_{k-1} **do**
10: N_k← "c1 ∧ c2"
11: **end for**
12: **while** N_k≠ { } **do**
13: N_k←N_k – { c | *ConjNumberPI* (N_k.condition1, N_k.condition2) ≤ *NumberPI*(N_k.condition1.attribute1, N_k.condition1.attribute2) **or** *ConjNumberPI*(N_k.condition1, N_k.condition2) ≤ *NumberPI*(N_k.condition2.attribute1, N_k.condition2.attribute2) }
14: N_k←N_k – { c | *ConjPIRatio* (N_k.condition1, N_k.condition2)<α **or** *ConjPIRatio*(N_k.condition1, N_k.condition2) > β }
15: *CC*←*CC* ∪ N_k
16: **for** c1 ∈ N_k **and** c2 ∈ N_k **do**
17: N_{k+1}← "c1 ∧ c2"
18: **end for**
19: k ← k+1
20: **end while**

Figure 4: Algorithm to discover the correlation set.

Heuristic 3. Atomic conditions are pruned when the process instance ratio (*PIRatio*) is under alpha or above beta (line 6). This heuristic checks that the partitioning of the future log does not only have one or two big instances, or many short instances. *PIRatio* (Eq. 5) is measured as the number of process instances (*NumberPI*) divided into non-null values for both attributes. In turn, *NumberPI* (Eq. 6) is heuristically assessed as the distinct attribute values for all the different couples of events (executed in a row) containing both attributes. This is the first difference, since the previous algorithm first calculates a set of correlated event pairs, and then it computes a recursive closure over that set (Motahari-Nezhad et al., 2011). In contrast, our algorithm estimates *NumberPI* by considering the number of possible pairs of correlated events. This change has been made taking into account that the recursive closure evaluation is time-consuming (the complexity of graph closure algorithms in literature is often $O(2^n)$ since they check each pair of nodes for the remaining of pairs). On the contrary, the expected results using this proposal can be considered as heuristic approximation with a lower computational cost (i.e., $O(n)$ since this technique only evaluates the list of event pairs). This is the first difference regarding the algorithm proposed by *Motahari-Nezhad et al.* (Motahari-Nezhad et al., 2011).

$$\boldsymbol{DistinctRatio}(\boldsymbol{a_i}) = \frac{distinct(a_i)}{nonNull(a_i, a_j)} \quad (1)$$

$$\boldsymbol{SharedRatio}(\boldsymbol{a_i}, \boldsymbol{a_j}) = \frac{distinct(a_i, a_j)}{max\left(distinct(a_i), distinct(a_j)\right)} \quad (2)$$

$$\boldsymbol{\alpha} = \frac{distinct_{MAX}(a_i)}{NumberOfEvents} \quad (3)$$

$$\boldsymbol{\beta} \epsilon\ [0.25, 1] \quad (4)$$

$$\boldsymbol{PIRatio}(\boldsymbol{a_i}, \boldsymbol{a_j}) = \frac{|NumberPI(a_i, a_j)|}{nonNull(a_i, a_j)} \quad (5)$$

$$\boldsymbol{NumberPI}(\boldsymbol{a_i}, \boldsymbol{a_j}) = \{v : \exists\, e, e' \epsilon Events | e.attribute1 = a_i \wedge e'.attribute2 = a_j \wedge v = e.attribute1.value = e'.attribute2.value \wedge e.timestamp > e'.timestamp\} \quad (6)$$

$$\boldsymbol{ConjPIRatio}(\boldsymbol{c_i}, \boldsymbol{c_j}) = \frac{|ConjNumberPI(c_i, c_j)|}{nonNull(c_i.a_1,\ c_i.a_2,\ c_j.a_1,\ c_j.a_2)} \quad (7)$$

$$\boldsymbol{ConjNumberPI}(\boldsymbol{c_i}, \boldsymbol{c_j}) = NumberPI(c_i.a_1, c_i.a_2) \cap NumberPI(c_j.a_1, c_j.a_2) \quad (8)$$

Moreover, *beta* (Eq. 4) which can be established between 0.25 and 1 is used as another threshold to evaluate the average length of the outgoing instances. For instance, a beta value of 0.5 (as is usually used) signifies that conditions leading to process instances with length above or equal to the half of the total events would be discarded.

After atomic conditions are filtered out, the algorithm (see Figure 4) builds all the possible conjunctive conditions based on the combination of outgoing atomic conditions (lines 7-11). These conditions are then pruned by applying two heuristics (lines 13-14). After that, new conjunctive conditions by combining the remaining previous conditions are iteratively evaluated (lines 15-19).

The first heuristic (line 13) applied to filter out conjunctive conditions is based on the monotonicity of the number of process instances. This is the second difference with regard to (Motahari-Nezhad et al., 2011), since this algorithm considers the number of process instances (but not the length of instances) to evaluate the monotonicity heuristic. This heuristic is based on the idea that the number of process instances for a conjunctive condition is

always higher than the number for their simpler conditions in isolation. Conjunctive conditions that do not increase the number of process instances are therefore pruned, since they are subsumed in their atomic conditions. The number of process instances obtained though conjunctive conditions (*ConjNumberPI)* (Eq. 8) is based on (Eq. 6), which is defined for simple conditions, and is measured by intersecting both component conditions of the conjunctive one.

```
Input: Events; AC: the set of atomic conditions; CC: the set of conjunctive conditions
Output: Log: the final event log
1: process; instance
2: for e1 ∈ Events ∧e2 ∈ Events ∧ e1.proces = e2.process
   ∧ e1.timestamp ≤ e2.timestamp do
3:    if e1.starting=true ∧ ∀n ∈Log.processes.name, process.name=n then
4:       process.name ← e1.process
5:    end if
6:    for ac ∈AC ∧cc ∈CCdo
7:       if ∃i, e1.attributes[ i ].name = ac.attribute1 ∧
         ∃i', e2.attributes[ i' ].name = ac.attribute2 ∧
         e1.attributes[ i ].value = e2.attributes[ i' ].value ∧
         ∃j, e1.attributes[ j ].name = cc.condition1.attribute1 ∧
         ∃j', e2.attributes[ j' ].name = cc.condition1.attribute2 ∧
         e1.attributes[ j ].value = e2.attributes[ j' ].value ∧
         ∃k, e1.attributes[ k ].name = cc.condition2.attribute1 ∧
         ∃k', e2.attributes[ k' ].name = cc.condition2.attribute2 ∧
         e1.attributes[ k ].value = e2.attributes[ k' ].value then
8:             instance.id ←e1.attributes[ i ].value + e2.attributes[ i' ].value +
               e1.attributes[ j ].value + e2.attributes[ j' ].value +
               e1.attributes[ k ].value + e2.attributes[ k' ].value
9:          instance.events←instance.events∪ {e1, e2}
10:            process.instances←process.instances∪ {instance}
11:         end if
12:      end for
13:      Log.processes←Log.processes∪ (Fluxicon Process Laboratories)
14:   end for
```

Figure 5: Algorithm to discover process instances.

The algorithm also applies the same heuristic about the partitioning of the log to the conjunctive conditions (line 14). Thereby, the ratio of process instances is also evaluated for conjunctive conditions (*ConjPIRatio*) (Eq. 7). When *ConjPIRatio* is under alpha or above beta threshold the conjunctive condition is discarded.

In conclusion, the proposed algorithm adapts the algorithms provided by (Motahari-Nezhad et al., 2011) thus adjusting to traditional systems. There are two main changes as seen above in this section: (i) the way in which the expected number of process instances is calculated for each condition; and (ii) the monotonicity heuristic which only takes into account the length of the estimated process instances.

3.3 Discovering Process Instances

The last stage, after obtaining the correlation set, attempts to discover process instances using the correlation set in order to build the final event log, which will be written following the MXML (Mining XML) format (Van der Aalst et al., 2009). MXML is a notation based on XML and is the most common format used by most process mining tools.

Figure 5 shows the algorithm to correlate all the events of the intermediate dataset in its own process instance within the event log. The algorithm explores all the candidate events pairs, i.e., those pairs that belong to the same process and which were executed in a row (line 2). When an event was recorded as the start point of a process, the target process takes this name (lines 3-5). For each candidate event pair, all the atomic and conjunctive conditions of the correlation set are evaluated (line 7). If the event pair meets all the conditions, then it is a correlated event pair and these events are then put into the respective process instance, and in turn, the instance is added to the process (lines 9-10). Process instances are previously identified by means of the specific values of the events' attributes involved in the correlation set (line 8). Each process found during the event pair exploration, together with all the discovered process instances, is finally added to the event log (line 13).

Business process can be subsequently discovered from the MXML event logs by applying different well-known techniques and algorithms developed from the business process mining field.

4 CASE STUDY

This section presents a case study conducted with a real-life information system. The whole set of artefacts involved in the study are online available in (Pérez-Castillo, 2012).

The object of the study is the proposed technique and the purpose of the study is to demonstrate the feasibility of the technique in terms of accuracy. The main research question therefore is:

MQ. - *Can the technique obtain correlation sets for generating event logs from a traditional system which could on their turn be used to discover the business processes supported by the system?*

Additionally, the study evaluates two secondary research questions:

AQ1. - *How well does this technique perform compared to the previously developed technique?*

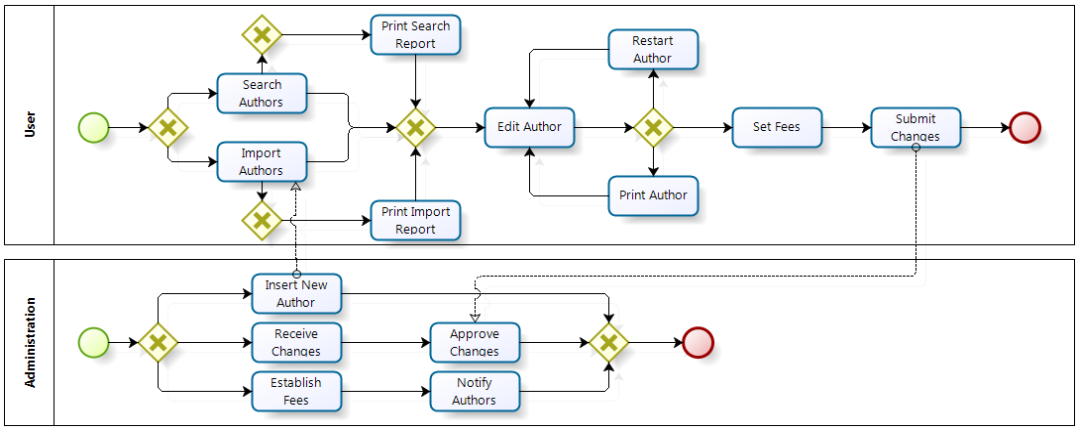

Figure 6: Reference business process model of the AELG-members system.

AQ2. - *How much time does the technique require to discover correlation sets regarding the size of datasets?*

AQ1 evaluates the gain of this new approach over the previous one employing an isolated source code object as correlation set. For the evaluation of this secondary question, the result of this study is compared with the result obtained in a previous study that validated the previous approach using the same traditional information system (Pérez-Castillo, Weber et al., 2011). AQ2 assesses the time spent on discovering correlations sets in order to know if the technique is scalable to a large dataset.

In order to answer the main research question, the study follows a qualitative research approach by comparing a reference model and the obtained one. The study first considers the business process model previously provided by business experts (the reference model). Secondly, the study obtains an event log (using the discovered correlation set) and compares the business processes instances collected in the log together with the reference business process model. The comparison between models evaluates the degree of conformance of the obtained model regarding the reference one, which is done by scoring the number of business activities in common with the reference business process model.

4.1 Case under Study

The traditional (non-process-aware) information system under study was *AELG-members*, which supports the administration of an organization of Spanish writers. From a technological point of view, *AELG-members* is a *Java* standalone application with an architecture that follows the traditional structure on three layers: (i) the *domain* layer supporting all the business entities and controllers; (ii) the *presentation* layer dealing with the user interfaces; and (iii) the *persistency* layer handling data access. The total size of the legacy system is 23.5 KLOC (thousands of lines of source code).

Figure 6 shows the business process supported by the system under study, which is considered as the reference business process model. The main business activities carried out by the writers' organization, including author registration, importing author information from different sources, cancellation of memberships, author information management and payment of fees.

4.2 Execution

For the execution of the case study, all the stages of the proposed technique were semi-automated by different tools. The steps carried out during the study execution were the following:

1. The AELG-members system was instrumented through the Event Traces Injector tool (Pérez-Castillo, 2012) which was modified to support the addition of candidate correlation attributes by experts. Six attributes were selected to be collected together with events (see Table 1). Some attributes regarding the identification of author were first selected due to the business processes focuses on this entity. Other attributes related to fees were also selected since experts expect process instances end when annual fees of an author are paid.
2. The instrumented version of AELG-members was normally executed in the same production environment. The execution consisted of storing events and candidate correlation attributes in a SQL Server 2005 database until significant datasets to conduct the study were collected. Three different sizes of dataset (above 2000, 7000 and 15000

events) were considered to test different configurations.

3. The algorithm for discovering the correlation set (Figure 4) was then applied to the datasets. Unlike in previous stages, this algorithm was implemented by means of a set of stored procedures using PL/SQL which executes a set of queries from datasets. Since the beta threshold (Eq. 4) can be chosen by business experts (Motahari-Nezhad et al., 2011), the algorithm was applied with four different values: 0.25, 0.5, 0.75 and 1. The four different correlation sets obtained for each configuration are shown in Table 2. An event log was obtained for each different correlation set by means of the algorithm presented in Figure 5, which was also implemented through PL/SQL procedures.

4. The four event logs were finally analyzed and compared with the reference model. For this purpose, the *ProM* tool (Van der Aalst et al., 2009) was used to discover the respective business process models for each log. The study particularly used the genetic mining algorithm of *ProM* since it is the most accurate one (Medeiros et al., 2007). Finally, the conformance of each business process model with the reference model was analyzed according to the aforementioned qualitative research approach (cf. Section 4.1).

Table 1: Candidate correlation attributes selected.

Attribute ID	Java Class	Output Method
1	FeeVO	getIdAuthor
2	AuthorVO	getId
3	AuthorVO	isHistoric
4	AuthorVO	getMemberNumber
5	PublicAuthorVO	getId
6	AuthorVO	getFees

Table 2: Correlation sets and time obtained in each case.

	Events	Correlation Attributes	β=0.25	β=0.5	β=0.75	β=1
Correlation Sets	2432	10412	A	C	C	C
	7608	33278	A	C	C	C
	15305	74136	B	C	D	D
Time (s)	2432	10412	12	15	16	15
	7608	33278	41	56	55	55
	15305	74136	113	150	151	147

Table 3: Correlation sets (numbers 1 to 5 refer to attribute ID of Table 1; letters o to s refer to atomic conditions).

Atomic Conditions	A	o : 1=1	p : 2=2	q : 4=4	r : 6=6	
	B	o : 1=1	p : 2=2	q : 4=4	r : 6=6	
	C	o : 1=1	p : 2=2	q : 4=4	r : 6=6	s : 5=5
	D	o : 1=1	p : 2=2	q : 4=4	r : 6=6	s : 5=5
Complex Conditions	A	o ∧ q	p ∧ q			
	B	o ∧ q	p ∧ q	r ∧ q	r ∧ p	
	C	o ∧ q	p ∧ q	r ∧ q	r ∧ p	
	D	o ∧ q	p ∧ q	r ∧ q	r ∧ p	o ∧ s

4.3 Analysis and Interpretation

Table 2 and Table 3 summarize the results obtained after cases study execution, showing the correlation sets (A, B, C and D) obtained for each combination of dataset (2432, 7608 and 15305 events) and beta value (0.25, 0.5, 0.75 and 1). Table 2 also shows the time spent on discovering each correlation set as well as the particular atomic and conjunctive conditions of each set.

After obtaining the corresponding event log and discovering the respective business process for the *AELG-Members* system, it was perceived that the most accurate correlation set was 'A'. The set 'A' leads to the business process with the highest conformance degree (93%). This means the business process discovered using set 'A' had the highest number of business activities in common with the reference business process model (13 from the 14 tasks).

The same conclusion can be stated by analyzing the conditions of correlation set 'A' (see Table 3). Set 'A' is less restrictive (compared to the other sets) and contains fewer correlation conditions. Despite this fact, it contains all the atomic conditions necessary to evaluate the identity of each writer (i.e., *getIdAuthor, getId* and *get MemberNumber*). Additionally, set 'A' also contains the atomic condition to know when a fee is paid (i.e., *getFees*), which signifies that a particular process instance ends for a writer.

Moreover, regarding complex conditions of correlation set 'A', there is a conjunctive condition linking *FeeVo.getIdAuthor* together with *AuthorVO.getId*, which signifies that the managed fees must correspond to the same writer of a particular process instance. Finally, set 'A' also works well because the categorical correlation attribute *AuthorVO.isHistoric* was properly discarded, since these kinds of attributes (e.g., Boolean variables) split the datasets into only two instances.

The remaining correlation sets (B, C and D) are similar to correlation set 'A', since all those sets contain all correlation conditions of 'A'. However, those sets incorporate more conditions, and although they provide alternative event correlations, they are more restrictive. This means that some process instances obtained using 'A' could be split in two or more instances in case sets B, C or D were used as the correlation set instead of set 'A'. These sets led to conformance values between 64% and 86%, which respectively correspond to 9 and 12 tasks in common with the 14 tasks of the reference model.

Regarding the evaluation of *AQ1,* in the previous case study with the same system (Pérez-Castillo et al., 2011), the Java class '*AuthorVO'* was selected as the classifier to collect correlation information during the system instrumentation stage. During system execution, the runtime values of the *AuthorVO* objects were used to correlate events. As a result, all the process instances in the event log were obtained with all the events regarding each writer. Unlike the current approach, not all the different executions of the reference business process (see Figure 6) for each author were detected. For example, every time a writer pays the annual fee, it should be detected as the end of a process instance. This kind of aggregation works by using any correlation set obtained with the current approach. However, as per the previous approach, not all the events of the same writer could be grouped into fine-grained process instances, since the sole information to correlate events was *AuthorVO* objects.

The conformance degree in the previous case study with the same system was 77% (Pérez-Castillo et al., 2011), while the degree obtained with the proposed technique is 93% (obtained with set 'A'). In fact, the business process obtained with the previous technique was complex and visually intricate due to several crossing sequence flows. As a result, *AQ1* can be positively answered.

In order to demonstrate the feasibility of the proposal, the time spent on discovering correlation sets was analyzed according to the additional question *AQ2*. Outlier beta values such as 1, and especially 0.25 lead to shorter times (see Table 2). This is due to the fact that outlier values allow the algorithm to quickly prune non-promising correlation sets, saving much time. Anyway, it should be noted that the time regarding the beta value is approximately linear. Besides, regarding the number of events, the time is non-linear. The time is lower for smaller datasets and higher for larger ones. It seems the trend of the time follows a quadratic function. This is due to the fact that every event must be checked for all the remaining events according to the proposed algorithm.

In conclusion, the main research question can be positively answered. This means that the technique is able to correlate events from traditional system, and in turn, it produces a gain regarding techniques previously developed. However, the time spent on discovering correlation sets is quadratic, and huge datasets may be time-consuming.

4.4 Evaluation of the Validity

The most important threat is the fact that the code could be poorly instrumented. The obtained results clearly depend on the candidate correlation attributes selected at the beginning of the study. If business experts select an incomplete or erroneous set of candidate correlation attributes, the outgoing results could be quite different. In order to mitigate this threat we propose repeating the study using an iterative approach in which experts can select or remove some candidate correlation attributes according to the results obtained for each iteration. This way, the list of candidate correlation attributes can be iteratively refined.

Moreover, correlation sets do not always have to be obtained under lower beta values (e.g., 0.25). A lower beta value often implies a more restrictive correlation set and vice versa. The beta threshold can therefore be established by business experts depending on the constraint degree to be applied to the particular set of candidate correlation attributes. This threat can be addressed by repeating the study with different cases and different beta values.

5 CONCLUSIONS

This paper presents a technique to discover the correlation set in order to generate event logs from traditional (non-process aware) information systems. This challenge is important in traditional systems since (i) they do not have any in-built mechanism to record events and (ii) captured events do not have any reference to the process instance they belong to.

The technique consist of three stages: (i) the selection of candidate correlation attributes and injection of statements into the source code to collect events during system execution; (ii) the discovery of the correlation set from collected events; and (iii) the generation of the final event logs by correlating events using the discovered correlation conditions.

All the stages of the technique are semi-automated, making it possible to validate the proposal by conducting a case study with a real-life system. The study demonstrates the feasibility of the technique to discover correlation sets that lead to well-formed event logs and the gain regarding previous techniques in terms of accuracy. The main implication of the results is that this technique contributes to the application of well-proven techniques and algorithms from the process mining field. So far, such business process mining

techniques work with event logs that are often obtained only from process-aware information systems.

The work-in-progress focuses on conducting another case study with a healthcare information system to obtain strengthened conclusions about empirical validation. Moreover, concerning the selection of candidate correlation attributes, a mechanism to analyse source code and provide business experts with some insights about the most appropriate attributes will be developed.

ACKNOWLEDGEMENTS

This work has been supported by the *FPU Spanish Program*; by the R+D projects funded by *JCCM*: ALTAMIRA (PII2I09-0106-2463), INGENIO (PAC08-0154-9262) and PRALIN (PAC08-0121-1374)*;* and MITOS (TC20091098) funded by the *UCLM*.

REFERENCES

Burattin, A. and R. Vigo (2011). A framework for Semi-Automated Process Instance Discovery from Decorative Attributes. *IEEE Symposium on Computational Intelligence and Data Mining (CIDM'11)* Paris, France: 176-183.

Castellanos, M., K. A. d. Medeiros, J. Mendling, B. Weber and A. J. M. M. Weitjers (2009). Business Process Intelligence. *Handbook of Research on Business Process Modeling*. J. J. Cardoso and W. M. P. van der Aalst, Idea Group Inc.: 456-480.

Ferreira, D. and D. Gillblad (2009). Discovering Process Models from Unlabelled Event Logs. Business Process Management. U. Dayal, J. Eder, J. Koehler and H. Reijers, Springer Berlin / Heidelberg. 5701: 143-158.

Fluxicon Process Laboratories (2009). XES 1.0 Standard Definitio (Extensible Event Stream). http://www.xes-standard.org/.

Hammoud, N. (2009). Decentralized Log Event Correlation Architecture. *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*. France, ACM: 480-482.

McGarry, K. (2005). "A Survey of Interestingness Measures for Knowledge Discovery." *Knowl. Eng. Rev.* 20(1): 39-61.

Medeiros, A. K., A. J. Weijters and W. M. Aalst (2007). "Genetic Process Mining: An Experimental Evaluation." *Data Min. Knowl. Discov.* 14(2): 245-304.

Motahari-Nezhad, H. R., R. Saint-Paul, F. Casati and B. Benatallah (2011). "Event Correlation for Process Discovery From Web Service Interaction Logs." *The VLDB Journal* 20(3): 417-444.

Myers, J., M. R. Grimaila and R. F. Mills (2010). Adding Value to Log Event Correlation Using Distributed Techniques. *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*. Oak Ridge, Tennessee, ACM: 1-4.

Pérez-Castillo, R. (2012). "Experiment Results about Assessing Event Correlation in Non-Process-Aware Information Systems " Retrieved 09/02/2012, 2012, from http://alarcos.esi.uclm.es/per/rpdelcastillo/CorrelationExp.html#correlation.

Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán and M. Piattini (2010). "Toward Obtaining Event Logs from Legacy Code." *Business Process Management Workshops (BPI'10)* Lecture Notes in Business Information Processing ((LNBIP 66 - Part 2)): 201–207.

Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán and M. Piattini (2011). "Generating Event Logs from Non-Process-Aware Systems Enabling Business Process Mining." *Enterprise Information System Journal* 5(3): 301–335.

Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán and M. Piattini (2011). "Process Mining through Dynamic Analysis for Modernizing Legacy Systems." *IET Software Journal* 5(3): 304–319.

Rozsnyai, S., A. Slominski and G. T. Lakshmanan (2011). Discovering Event Correlation Rules for Semi-Structured Business Processes. *Proceedings of the 5th ACM international conference on Distributed event-based system*. New York, USA, ACM: 75-86.

van der Aalst, W. and A. J. M. M. Weijters (2005). Process Mining. Process-aware information systems: bridging people and software through process technology. M. Dumas, W. van der Aalst and A. Ter Hofstede, John Wiley & Sons, Inc.: 235-255.

Van der Aalst, W. M. P., B. F. Van Dongenm, C. Günther, A. Rozinat, H. M. W. Verbeek and A. J. M. M. Weijters (2009). ProM: The Process Mining Toolkit. *7th International Conference on Business Process Management (BPM'09)* - Demonstration Track. Ulm, Germany, Springer-Verlag. 489: 1-4.

Weske, M. (2007). Business Process Management: Concepts, Languages, Architectures. Leipzig, Germany, Springer-Verlag Berlin Heidelberg.

Zou, Y. and M. Hung (2006). An Approach for Extracting Workflows from E-Commerce Applications. *Proceedings of the Fourteenth International Conference on Program Comprehension, IEEE Computer Society*: 127-136.