Enhancing Declare Maps Based on Event Correlations

R.P. Jagadeesh Chandra Bose[1], Fabrizio M. Maggi[2] and Wil M.P. van der Aalst[1]

[1] Eindhoven University of Technology, The Netherlands.
[2] University of Tartu, Estonia.

Abstract. Traditionally, most process mining techniques aim at discovering *procedural* process models (e.g., Petri nets, BPMN, and EPCs) from event data. However, the variability present in less-structured flexible processes complicates the discovery of such procedural models. The "open world" assumption used by declarative models makes it easier to handle this variability. However, initial attempts to *automatically discover declarative process models* result in cluttered diagrams showing misleading constraints. Moreover, additional data attributes in event logs are not used to discover meaningful causalities. In this paper, we use *correlations* to prune constraints and to disambiguate event associations. As a result, the discovered process maps only show the more meaningful constraints. Moreover, the data attributes used for correlation and disambiguation are also used to find *discriminatory patterns*, identify *outliers*, and analyze *bottlenecks* (e.g., when do people violate constraints or miss deadlines). The approach has been implemented in ProM and experiments demonstrate the improved quality of process maps and diagnostics.

1 Introduction

Processes executed in today's world are often supported and controlled by information systems, which record events, like messages and transactions, in so-called *event logs*. *Process mining* aims at discovering, monitoring and improving real-life processes by extracting knowledge from event logs. *Process discovery*, *conformance checking*, and *process enhancement* are three main process mining tasks [3]. In particular, process enhancement aims at enriching and extending existing process models with information retrieved from logs, e.g., a process model can be extended with performance-related information such as flow time and waiting time.

Choosing a suitable representational bias for process discovery, visualization, and analysis is one of the challenges in process mining [11]. Process characteristics play a significant role in the selection of a suitable representational bias. Processes working in stable environments are typically highly predictable, i.e., it is easy to determine in advance the way how processes execute and behave (e.g., a process for handling travel requests). Procedural languages, such as BPMN, UML ADs, EPCs, and Petri nets, are suitable for describing such processes because it is easy to explicitly represent all allowed behavior of the process at hand [4, 22]. In contrast, processes operating in flexible/turbulent environments are often more complex and less predictable. Here, process participants make decisions based on multiple (possibly conflicting) objectives and have a lot of freedom in the process execution (e.g., a doctor in a healthcare process).

Declarative process modeling languages like *Declare* [4] are more suitable for such environments. Declarative models describe a process as a list of constraints that must be satisfied during the process execution. In declarative languages, an "open world" is assumed where everything is allowed unless it is explicitly forbidden. Declarative models are widely used in domains and applications where processes cannot be "straightjacketed" into a procedural model [7, 15, 16, 25].

Declare is a declarative language introduced in [4] that combines a formal semantics grounded in Linear Temporal Logic (LTL) with a graphical representation for users.[3] A *Declare map* is a set of Declare constraints each one with its own graphical representation and LTL semantics (see [4] for a full overview of Declare). In recent years, approaches to discover Declare models from event logs [17–19] and approaches to check conformance of Declare models with respect to event logs [9, 13] have been proposed.

Although promising, these approaches face several challenges and limitations when being applied to real-life event logs. First of all, discovery approaches typically generate too many constraints resulting in incomprehensible Declare maps. Most of the generated constraints have no domain significance and are uninteresting for experts/analysts. Second, when evaluating the satisfaction of a constraint, one often faces ambiguities in connecting events that "activate" the constraint (activations) and events that "fulfill" it (target events). For example, consider trace $\mathbf{t} = \langle A, A, B, B \rangle$ and the constraint *response(A,B)*.[4] It is unclear which instance of activity B can be associated to the two instances of activity A. Such ambiguities inhibit a correct evaluation of constraints in terms of satisfaction/violation and the application of certain types of analysis such as performance analysis (e.g., computing the response time of constraints).

One of the main reasons for these ambiguities and incomprehensible maps is the exclusive focus on the *control-flow* perspective. Typically, event logs contain additional information in the form of attributes and values. Let $\mathbf{t} = \langle A(x = 1, y = 2), A(x = 2, y = 1), B(x = 2, y = 0), B(x = 1, y = 4) \rangle$ be the above trace with its data attributes. The additional information suggests that the first instance of A is related to the second instance of B and the second instance of A is related to the first instance of B because they share the same value for attribute x.

In this paper, we propose an approach to automatically discover significant event correlations between events involved in a constraint and use these correlations to (i) enhance (annotate) a discovered Declare map and improve its comprehensibility (see Fig. 1), (ii) prune discovered constraints that are uninteresting, (iii) disambiguate events so that correct events are correlated, (iv) extend a Declare map with meaningful performance information, and (v) improve the diagnostic abilities by finding discriminatory patterns (if any) between different classes of behavior (e.g., patterns that may discriminate between conformant and non-conformant activations of a constraint).

We evaluate the proposed approach using a real-life event log provided for the 2011 BPI Challenge [2], which pertains to the treatment of patients diagnosed with cancer in a large Dutch academic hospital. Fig. 2 depicts the Declare maps obtained using the hospital's event log for the *response* and *precedence* constraints with and without correlations (the correlations used are A.org:group = B.org:group and A.Producer

[3] In the remainder, LTL refers to the version of LTL tailored towards finite traces.

[4] *response(A,B)* = If A *occurs*, then eventually B *follows* after A.

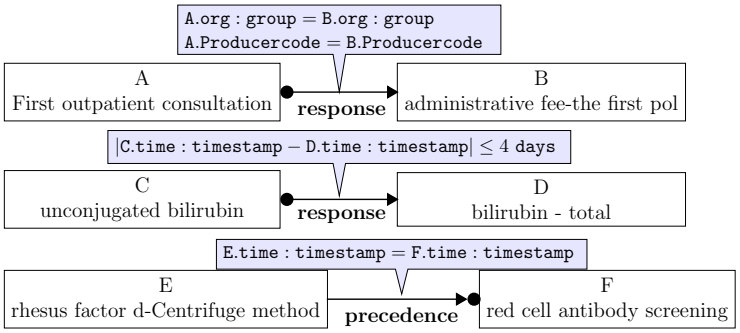
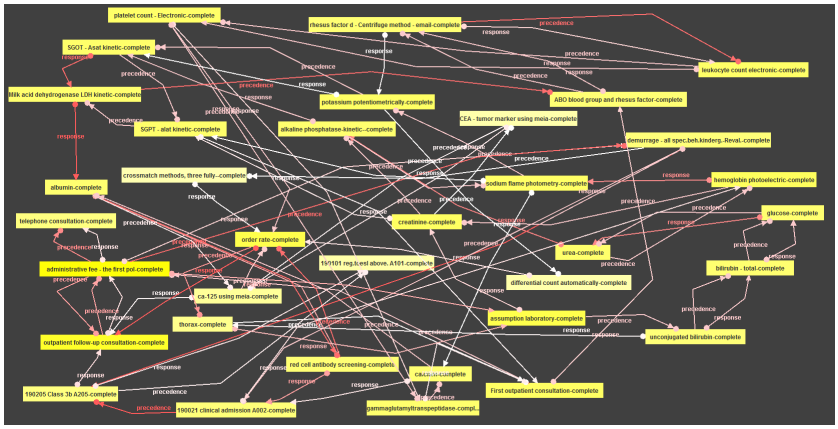
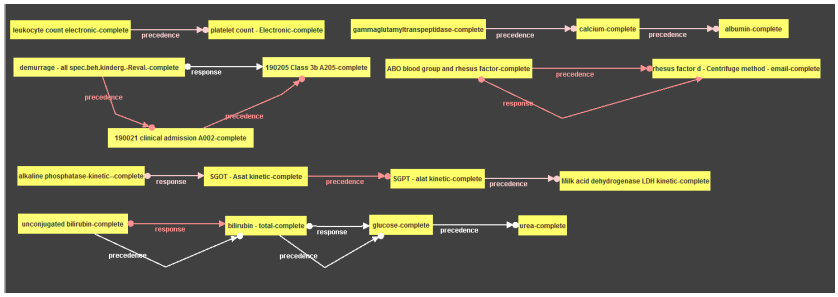


Fig. 1. A Declare map annotated with correlations.



(a) without correlations



(b) with correlations

Fig. 2. Declare maps discovered for a hospital's event log [2] using the response and precedence constraints with and without correlations.

code = B.Producer code). We can clearly see that the map obtained using correlations (Fig. 2(b)) is much simpler and more comprehensible than the map using the conventional approach (Fig. 2(a)).

The remainder of this paper is organized as follows. Section 2 presents some preliminaries on event logs and the Declare language. Section 3 presents some of the issues in contemporary approaches in the discovery of Declare maps and highlights how correlations can help address them. Section 4 presents our approach to discovering event correlations and discriminatory patterns between different classes of constraint behavior. Section 5 presents and discusses the experimental results. Related work is presented in Section 6. Finally, Section 7 concludes the paper.

2 Preliminaries

In this section, we introduce some preliminary notions. In particular, in Section 2.1, we summarize what an event log is and, in Section 2.2, we give an overview of the Declare language.

2.1 Event Logs

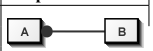
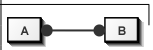
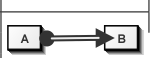
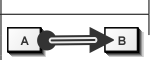
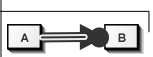
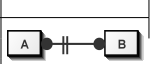
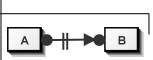
An event log captures the manifestation of events pertaining to the instances of a single process. A *process instance* is also referred to as a *case*. Each event in the log corresponds to a single case and can be related to an activity or a task. Events within a case need to be *ordered*. An event may also carry optional additional information like *timestamp*, *transaction type*, *resource*, *costs*, etc. For analysis, we need a function that maps any event e onto its *class* $\overline{e}$. In this paper, we assume that each event is classified based on its activity. We use the following notations:

- $\mathcal{A}$ denotes the set of *activities*. $\mathcal{A}^+$ is the set of all non-empty finite sequences of activities from $\mathcal{A}$.
- A *process instance* (i.e., a case) is described as a *trace* over $\mathcal{A}$, i.e., a finite sequence of activities. Examples of traces are $\mathbf{t}_1 = \langle A, B, C, D\rangle$ and $\mathbf{t}_2 = \langle A, B, B, B, A, D\rangle$.
- Let $\mathbf{t} = \langle \mathbf{t}(1), \mathbf{t}(2), \ldots, \mathbf{t}(n)\rangle \in \mathcal{A}^+$ be a trace over $\mathcal{A}$. $|\mathbf{t}| = n$ denotes the *length* of trace $\mathbf{t}$. $\mathbf{t}(k)$ represents the k^{th} activity in the trace.
- An *event log*, $\mathcal{L}$, corresponds to a *multi-set* (or bag) of traces from $\mathcal{A}^+$. For example, $\mathcal{L} = [\langle A, B, C, D\rangle, \langle A, B, C, D\rangle, \langle A, B, B, B, A, D\rangle]$ is a log consisting of three cases. Two cases follow trace $\langle A, B, C, D\rangle$ and one case follows trace $\langle A, B, B, B, A, D\rangle$.

2.2 Declare: Some Basic Notions

Declare is a declarative process modeling language introduced by Pesic and van der Aalst in [4]. A *Declare map* is a set of constraints that must hold in conjunction during the process execution. Declare constraints are equipped with graphical notations and LTL semantics. The most frequently used Declare constraints are shown in Table 1. However, the language is extensible and new constraints can be added by providing a graphical representation and corresponding LTL semantics. The results discussed in this paper only refer to positive relation constraints and not to negative relations (last three rows in Table 1). Indeed, since negative relation constraints forbid the occurrence of events, it is less natural to define the notion of correlation for these constraints.

Table 1. Graphical notation and textual description of some Declare constraints.

Constraint	Meaning	LTL semantics	Graphical notation
responded existence(A,B)	if A occurs then B occurs before or after A	$\Diamond A \rightarrow \Diamond B$	
co-existence(A,B)	if A occurs then B occurs before or after A and vice versa	$\Diamond A \leftrightarrow \Diamond B$	
response(A,B)	if A occurs then eventually B occurs after A	$\Box(A \rightarrow \Diamond B)$	
precedence(A,B)	if B occurs then A occurs before B	$(\neg B \sqcup A) \vee \Box(\neg B)$	
succession(A,B)	for A and B both precedence and response hold	$\Box(A \rightarrow \Diamond B) \wedge (\neg B \sqcup A) \vee \Box(\neg B)$	
alternate response(A,B)	if A occurs then eventually B occurs after A without other occurrences of A in between	$\Box(A \rightarrow \bigcirc(\neg A \sqcup B))$	
alternate precedence(A,B)	if B occurs then A occurs before B without other occurrences of B in between	$((\neg B \sqcup A) \vee \Box(\neg B)) \wedge \Box(B \rightarrow \bigcirc((\neg B \sqcup A) \vee \Box(\neg B)))$	
alternate succession(A,B)	for A and B both alternate precedence and alternate response hold	$\Box(A \Rightarrow \bigcirc(\neg A\, U\, B)) \wedge (((\neg B \sqcup A) \vee \Box(\neg B)) \wedge \Box(B \rightarrow \bigcirc((\neg B \sqcup A) \vee \Box(\neg B))))$	
chain response(A,B)	if A occurs then B occurs in the next position after A	$\Box(A \rightarrow \bigcirc B)$	
chain precedence(A,B)	if B occurs then A occurs in the next position before B	$\Box(\bigcirc B \rightarrow A)$	
chain succession(A,B)	for A and B both chain precedence and chain response hold	$\Box(A \rightarrow \bigcirc B) \wedge \Box(\bigcirc B \rightarrow A)$	
not co-existence(A,B)	A and B cannot occur together	$\neg(\Diamond A \wedge \Diamond B)$	
not succession(A,B)	if A occurs then B cannot eventually occur after A	$\Box(A \rightarrow \neg(\Diamond B))$	
not chain succession(A,B)	if A occurs then B cannot occur in the next position after A	$\Box(A \rightarrow \bigcirc(\neg B))$	

Consider the *response* constraint $\Box(A \rightarrow \Diamond B)$. This constraint indicates that if A *occurs*, B must eventually *follow*. Therefore, this constraint is satisfied for traces such as $\mathbf{t}_1 = \langle A, A, B, C \rangle$, $\mathbf{t}_2 = \langle B, B, C, D \rangle$, and $\mathbf{t}_3 = \langle A, B, C, B \rangle$, but not for $\mathbf{t}_4 = \langle A, B, A, C \rangle$ because $\mathbf{t}_4(3)$, i.e., the second instance of A, is not followed by a B. Note that, in $\mathbf{t}_2$, the response constraint is satisfied in a trivial way because A never occurs. In this case, we say that the constraint is *vacuously satisfied* [12]. In [9], the authors introduce the notion of *behavioral vacuity detection* according to which a constraint is non-vacuously satisfied in a trace when it is activated in that trace. An *activation* of a constraint in a trace is an event whose occurrence imposes, because of that constraint, some obligations on other events in the same trace. For example, A is an activation for the response constraint because the execution of A forces B to be executed eventually.

An activation of a constraint results in either a *fulfillment* (the obligation is met) or a *violation* (e.g., A is not followed by B in a response constraint). A trace is perfectly

compliant if there are no violations. Consider, again, the response constraint. In trace $\mathbf{t}_1$, the constraint is activated and fulfilled twice, whereas, in trace $\mathbf{t}_3$, the same constraint is activated and fulfilled only once. When a trace is not compliant w.r.t. a constraint, at least one activation leads to a violation. In trace $\mathbf{t}_4$, for example, the response constraint is activated twice (at $\mathbf{t}_4(1)$ and $\mathbf{t}_4(3)$): the activation at $\mathbf{t}_4(1)$ leads to a fulfillment (eventually B occurs), but the activation at $\mathbf{t}_4(3)$ leads to a violation (B does not occur subsequently). An algorithm to discriminate between fulfillments and violations for a constraint in a trace is presented in [9].

3 Correlations as a Means of Enhancing Declare Maps

Techniques for the automated discovery of Declare maps from event logs have been proposed in [17–19]. These approaches, although promising, typically generate *maps with too many constraints, have difficulties in correctly associating events, and do not provide diagnostic information.* This can be attributed to the fact that these techniques exploit only the control-flow perspective. Several of today's event logs contain rich information in the form of (event) attributes pertaining to the *data*, *resource*, and *time* perspectives.

In this paper, we advocate the use of these additional perspectives and investigate the correlations between event attributes as a means of addressing some of the above mentioned issues. Correlations are defined over event attributes and linked through *relationship operators* between them. For example, two events are correlated if they act upon common data elements of the process or if they are executed by the same resource etc. Such correlations can be used in conjunction to the control-flow relationship between events (defined in the form of Declare constraints) to further assess the relevance/significance of a constraint. Correlations can help us to:

- *prune uninteresting constraints:* we conjecture that constraints involving activities are interesting from a domain point of view only in cases where they share some common (data) elements of a process. For example, consider an insurance claim process where, apart from the handling of a claim application, applicants are asked to fill out a regular questionnaire. Clearly, in this process, the portion soliciting feedback does not interfere with the claim handling. Subsequently, the control-flow constraints between the activities involved in the claim handling and the activities involved in the questionnaire handling are less interesting to experts. This might be reflected in the activities in these two portions of the process sharing no or very few attributes (and thereby there are not significant correlations between them). Pruning such constraints will help reduce the number of uncovered constraints and improve the comprehensibility of a Declare map.
- *disambiguate events:* event associations that are ambiguous purely from a control-flow point of view can be disambiguated with additional conditions on their attributes. For example, consider trace $\mathbf{t}_1 = \langle A, B, C, B \rangle$ and the *response* constraint $\Box(A \rightarrow \Diamond B)$. Let us assume that activities A and B have a common attribute x and that we have an additional condition $A.x = B.x$ correlating these attributes for this constraint, i.e., the constraint now reads as "if A occurs, then B eventually follows and the value of attribute x is the same for both A and B". Now let us

assume that $\mathbf{t}_1(1).x = 1$, $\mathbf{t}_1(2).x = 2$, and $\mathbf{t}_1(4).x = 1$. Using the correlation, we can now clearly identify that the instance of B at $\mathbf{t}_1(4)$ is the one to be associated to the activation at $\mathbf{t}_1(1)$. Disambiguation of events facilitates a correct association of events involved in a constraint and thereby helps in performance analysis of a process (e.g., computing the response time more accurately).

– *improve diagnostic capabilities:* event correlations can be used for a plethora of diagnostic insights on process execution. One may use the discovered correlations to identify any potential exceptional executions/outliers. For example, let us assume that the correlation $A.x = B.x$ holds for 98% of the fulfillments of a response constraint $\Box(A \rightarrow \Diamond B)$. The 2% of the activations where the correlation does not hold (but considered as fulfillments purely from a control-flow perspective) may potentially be outliers or can be considered as a fulfillment due to wrong association of events for the constraint. Similarly, one may try to find if any discriminatory correlation patterns exist between different classes of behavior, e.g., between activations that are fulfillments and activations that are violations. For example, in an insurance claim, one may learn that a constraint is violated if the claim amount is greater than 1000 euros.

Furthermore, correlations can be used in defining conceptual groupings of activities. Different correlations between events can be used to define different conceptual groupings. For example, one may define equivalence classes based on a common attribute and consider all activities in that equivalence class as one conceptual group, e.g., the activities involving all events that are executed within the same department can be defined as one conceptual group. Such conceptual groupings of activities can be used for guiding the discovery of Declare maps towards results that are more significant from an application domain point of view [18].

For a categorization of correlations we refer to [6]. In this paper we use:

– *Property-based correlation*, i.e., events are classified based on a function operating on their attributes. For example, all claim applications referring to an amount greater than 1000 euros are grouped together.
– *Reference-based correlation*, i.e., two events are correlated if an attribute of the first event (identifier attribute) and an attribute of the second event (reference attribute) have the same value.
– *Moving time-window correlation*, i.e., two events are correlated if they occur within a given duration of one another (e.g., one hour).

We use an extended definition of reference-based correlation according to which two events are correlated if there is a function connecting an attribute of the first event with an attribute of the second event. This function can include not only equality but also operators such as *greater than*, *less than*, and *not equal to*. For example, an event of producing a document is correlated to an event of checking it if the resource that produces the document is different from the resource that checks it.

4 Discovering Correlations from Event Logs

Correlations can be provided by a domain expert, or alternatively, one can try to learn these correlations automatically from event logs. In this section, we focus on the auto-

mated discovery of correlations and discriminatory (correlation) patterns between different classes of behavior from event logs.

The XES standard [1] for event logs allows for events having attributes. XES supports data types such as string, date, boolean, int, and float (henceforth, we consider int and float types as *continuous*). Depending on the type, standard operators are supported. For example, we use the $\leq, \geq, <, >, =, \neq$ operators for continuous attributes and $=, \neq$ for string and boolean attributes. Timestamp (date) attributes are related using *before*, *after* operators in addition to all relation operations (i.e., $\leq, \geq, <, >, =, \neq$) over the time *difference* between two events.

We are interested in correlations between *comparable* attributes of different events, e.g., in an insurance claim process, attribute *amount claimed* is comparable to *amount issued*, but not to, say, *location*. If a priori knowledge about the domain is available, we can use that knowledge to identify/group attributes that are comparable. In the absence of prior domain knowledge, we consider *attributes having the same data type* to be comparable. Standard event attributes in XES are handled differently, e.g., although the attributes *concept:name* and *org:group* are of string type, they are not comparable.

Using the above correlation notion, we generate all feasible correlations for a given constraint. For discovering significant correlations, we partition the constraint activations into *ambiguous* activations and *non-ambiguous* activations. The definition of what constitutes an ambiguous activation is specific for each constraint type. We consider a fulfilled activation as non-ambiguous if there is only one possible target that can be associated to it. For example, for the *response* constraint $\Box(A \rightarrow \Diamond B)$, the activations in traces $\mathbf{t}_1 = \langle A, C, B \rangle$ and $\mathbf{t}_2 = \langle A, A, C, B \rangle$ are non-ambiguous whereas the activations in traces $\mathbf{t}_3 = \langle A, B, C, B \rangle$ and $\mathbf{t}_4 = \langle A, A, B, B \rangle$ are ambiguous. One may argue that the activations in trace $\mathbf{t}_2$ are also ambiguous because B can be associated to either of the two A's. We consider the scenario in $\mathbf{t}_1$ as *strongly non-ambiguous* and the scenario in $\mathbf{t}_2$ as *weakly non-ambiguous*. For each feasible correlation, we evaluate its *support* considering only non-ambiguous activations. The support of a correlation is defined as *the ratio between the number of activations in which that correlation is true and the total number of non-ambiguous activations*. We consider a feasible correlation as *significant* if its support is greater than a (user-specified) threshold. For correlations involving an attribute and a constant value, e.g., $B.timestamp - A.timestamp < \delta$, δ is derived based on the mean μ and standard deviation σ time difference of all non-ambiguous activations (for example δ can be set to $\mu + \sigma$).

Significant correlations thus discovered from non-ambiguous activations of a constraint can then be used to address the issues highlighted before, e.g., to disambiguate ambiguous activations. For each significant correlation, its *degree of disambiguation* is defined as *the ratio between the number of ambiguous activations that can be disambiguated and the total number of ambiguous activations*. Furthermore, different correlations can be combined using conjunctions or disjunctions to form complex correlations. Fig. 3 depicts the block diagram of discovering constraint correlations.

Discovering Discriminant Correlations An event log may exhibit several classes of behavior. For example, certain activations of a constraint may be eventually fulfilled while others may not. As another example, one may observe differences in the response

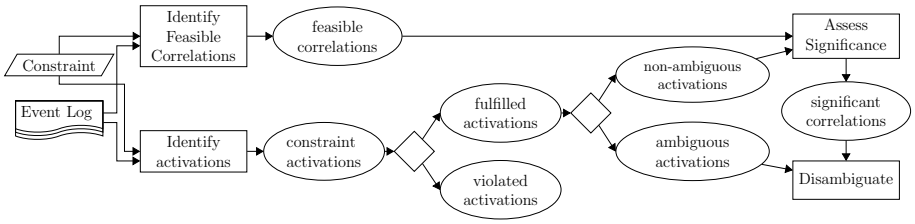


Fig. 3. Correlations are discovered for non-ambiguous activations and are subsequently used to disambiguate other (ambiguous) activations.

time for different activations of a constraint (one may distinguish the activations into *slow*, *medium*, and *fast* based on their response time). Such differences in behavior may be attributed to some of the characteristics of the events/traces, e.g., one may perceive differences in the response time of a constraint based on the resources involved in the execution of the activities or based on the attribute values (such as the claim amount or geography in a loan handling process). An analyst would be interested in uncovering any significant discriminatory correlations that can explain the different classes of behavior among the activations. We find such discriminatory correlations using the following three classification steps:

Step 1: Class Labeling. First, we select all the activations of a constraint and associate a class label to them. Different strategies for labeling can be adopted. For example, one can classify them as *conformant* or *non-conformant* based on whether they correspond to a fulfillment or to a violation. One can also consider all the *fulfilled* activations of a constraint and classify them as *slow*, *medium*, and *fast* based on their response time.

Step 2: Feature Extraction and Selection. The attributes of the events involved in a constraint and the process instance (case) attributes are considered as primitive features for finding discriminatory patterns. If all activations of a constraint (i.e., both fulfilled and violated) are selected in the previous step, then we only consider the correlations between attributes of the activations. If only fulfilled activations of a constraint are selected, then the correlations between attributes of the activations and attributes of the target events are also considered. This is due to the fact that a correlation involving an attribute of an activation and a target event can only be defined if the constraint is fulfilled (only in this case both activation and target event occur).[5]

Let $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ be the set of feasible correlations for a constraint, $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ be the set of attributes of the activation of a constraint, and $\mathcal{P} = \{p_1, p_2, \ldots, p_k\}$ be the set of case attributes corresponding to an activation. Each activation of a constraint can be translated into a vector where the dimensions correspond to $\mathcal{C} \cup \mathcal{A} \cup \mathcal{P} \cup \{C_l\}$; C_l is a special feature called the class label. The values of the features for each activation correspond to:

[5] This is true only for positive relation constraints (which are the ones considered in this paper). For negative relations the opposite applies (i.e., a correlation involving an attribute of an activation and a target event can only be defined if the constraint is violated because only in this case both activation and target event occur).

– $\{$*true, false*$\}$ if the feature is a correlation feature. The value is *true* if the correlation holds in the activation and *false* if it does not hold,
– the value of the attribute in the event corresponding to the activation of the constraint if the feature corresponds to $a_i \in \mathcal{A}$,
– the value of the case attribute corresponding to the process instance of the activation if the feature corresponds to $p_i \in \mathcal{P}$, and
– the class label of the activation if the feature is the class label.

The set of all activations upon transformation into a vector space can be seen as a dataset as depicted in Table 2.

Table 2. A labeled dataset defined by features.

Activation	c_1	c_2	...	c_n	a_1	a_2	...	a_m	p_1	p_2	...	p_k	C_l
1	true	false	...	true	50	xyz	...	1.5	1000	p	...	r	conformant
2	true	false	...	false	110	abc	...	3.25	500	q	...	s	non-conformant
3	false	true	...	true	64	ted	...	0.2	275	p	...	t	non-conformant
4	false	true	...	true	15	xyz	...	0.87	1255	u	...	s	conformant
⋮			⋮				⋮				⋮		

Step 3: Discovering Discriminatory Patterns. Given a dataset as depicted in Table 2, the goal of this step is to discover the patterns over the features, which are strongly correlated to the class label (e.g., conformant and non-conformant). We adopt standard data mining techniques, i.e., decision tree learning [23] and association rule mining [5, 14]. For the association rule mining, we adopt the special subset called the *class association rules* [14], which is an integration of classification rule mining and association rule mining. The details of these algorithms are beyond the scope of this paper. The result of this step are rules such as:

If $c_n = true$ AND $a_1 \leq 50$ AND $a_2 =$ xyz AND $p_1 \geq 1000$ then *conformant*;
If $a_1 \geq 60$ AND $a_2 \neq$ xyz AND $p_1 \leq 500$ then *non-conformant*.

Each rule can be associated with metrics such as the number of true positives (TP), false positives (FP), support and confidence. The quality of the entire set of discriminatory patterns uncovered can be assessed using standard metrics such as *accuracy*, *sensitivity*, *specificity*, *precision*, and *F1-score*.

5 Experiments and Results

The concepts presented in this paper have been implemented as the Extend Declare Map with Correlations and Extend Declare Map with Time Information plug-ins in ProM[6]. The former deals with the discovery and evaluation of correlations while the latter deals with performance analysis of Declare constraints (e.g., computing the

[6] ProM is an extensible framework that provides a comprehensive set of tools/plug-ins for the discovery and analysis of process models from event logs. See `www.processmining.org` for more information and to download ProM.

response times). The plug-ins take a Declare map and an event log as input and produce an enhanced Declare map annotated with data correlations and/or performance information. The input Declare map can either be discovered using the Declare Maps Miner plug-in or provided by a domain expert.

We have applied the proposed approach to the BPI challenge 2011 event log [2] pertaining to the treatment of patients diagnosed with cancer in a large Dutch academic hospital. The event log contains 1143 cases and $150,291$ events distributed across 623 event classes (activities). The event log contains domain specific attributes, e.g., *Producer code*, *Section*, *Activity code*, *Number of executions*, and *Specialism code* in addition to the standard XES attributes for events: *concept:name*, *lifecycle:transition*, *time:timestamp*, and *org:group*. We considered attributes with the same name to be *comparable* (i.e., an attribute x of the activation event is comparable only to attribute x of the target event) and explored the feasible correlations for various attributes.

We first generated a Declare Map from this event log using the Declare Maps Miner plug-in and considered constraints with a support of 50%. Activations that are fulfillments are further partitioned into ambiguous and non-ambiguous activations. Table 3 depicts the number of ambiguous and non-ambiguous activations for some constraints. Using the non-ambiguous activations we evaluated the support for the various correlations. Some significant correlations are depicted in Table 3 (refer columns correlation and support (correl.)).

From the table, we can see that for the response constraint $\Box(A \rightarrow \Diamond B)$ (where A corresponds to *First outpatient consultation* and B corresponds to *administrative fee - the first pol*), there are 559 ambiguous activations. Correlation *A.org:group = B.org:group* (i.e., both activities A and B are performed in the same department) holds for 94% of the 517 non-ambiguous activations. It is expected that the fee is decided and collected by the same department that performed the activation activity. However, it is interesting to see that 6% of the activations do not satisfy this correlation. It could be the case that by considering only the control-flow perspective, we have wrongly associated some administrative fee events thereby incorrectly evaluating the constraint as fulfillment for these activations. This correlation is able to disambiguate 57.96% of the 559 ambiguous activations. There exists another correlation *A.Producer code = B.Producer code* for this constraint, whose support is 93.61% in the non-ambiguous activations. This correlation is able to disambiguate 61.53% of the ambiguous activations.

For the response constraint $\Box(C \rightarrow \Diamond D)$ (where C corresponds to *unconjugated bilirubin* and D corresponds to *bilirubin - total*), we discover the correlation $|$ *C.time:timestamp* $-$ *D.time:timestamp*$| \leq 4$ days (i.e., activity D should be performed within 4 days of performing activity C). This event log exhibits coarse granular timestamps (recorded at the level of a day). The threshold of 4 days corresponds to $\mu + \sigma$ where μ and σ correspond to the mean and standard deviation time difference for all non-ambiguous activations of this constraint. This correlation holds in 99.63% of the non-ambiguous activations. The remaining 0.37% are most likely *outliers*. This correlation is able to disambiguate 81.61% of the ambiguous activations.

As another example, for the precedence constraint $(\neg F \sqcup E) \vee \Box(\neg F)$ (where E corresponds to *rhesus factor d - Centrifuge method* and F corresponds to *red cell antibody screening*), there are 603 and 932 non-ambiguous and ambiguous activations respec-

Table 3. Correlations discovered for some constraints and their support and degree of disambiguation. The encoded activities correspond to A = First outpatient consultation, B = administrative fee - the first pol, C = unconjugated bilirubin, D = bilirubin - total, E = rhesus factor d - Centrifuge method, F = red cell antibody screening.

constraint	support (constr.) (%)	#non-ambig. inst.	#ambig. inst.	**correlation**	support (correl.) (%)	degree of disambiguation(%)
response (A,B)	57.39	517	559	A.org:group = B.org:group	94.00	57.96
				A.Producer code = B.Producer code	93.61	61.53
response (C,D)	52.40	542	359	|C.time:timestamp − D.time:timestamp| ≤ 4 days	99.63	81.61
precedence (E,F)	54.85	603	932	E.time:timestamp = F.time:timestamp	100.00	96.45

tively. We discover that the correlation *E.time:timestamp = F.time:timestamp* holds in all the non-ambiguous activations (i.e., both these activities are performed on the same day). Using this correlation, we are able to disambiguate 96.45% of the ambiguous activations.

Although we discussed the applicability of correlations in disambiguation for the response and precedence templates, correlations exhibit a similar behavior for other templates too. Table 4 depicts the average and maximum degree of disambiguation across various constraints (with a support of 50%) for different templates. From the table, we can see that the approach proposed above is able to assist in disambiguation significantly.

Table 4. Degree of disambiguation for different templates.

Template	**#Constraints**	**Avg #Activations per constraint**		**Deg. of Disamb.**	
		non-ambi.	**ambi.**	**Avg. (%)**	**Max. (%)**
response	86	402	1321	51.68	95.76
precedence	250	842	1536	32.17	96.45
alternate response	53	733	601	70.67	100.00
alternate precedence	52	807	715	41.86	100.00
responded existence	584	682	2365	20.52	97.62

The discovered correlations can be used to reassess the fulfillment of constraint activations. For example, a response constraint $\Box(A \rightarrow \Diamond B)$ can be compounded with a correlation condition, *A.org:group = B.org:group* (i.e., in addition to B eventually following A, it is also required that they are executed by the same resource/department for an activation to be considered as fulfilled). Some activations that were deemed to be fulfilled when considering only the control-flow perspective, may no longer be fulfilled thereby impacting the *support* of the constraint, whose value, if less than a threshold, renders the constraint insignificant and a candidate for pruning. Table 5 illustrates how

correlations assist in pruning constraints. The first row in each constraint type depicts the number of constraints for varying support thresholds and without considering correlations, e.g., 371 response constraints have a support of at least 30% in the event log. The subsequent rows show the effect of adding correlations. For example, by adding a correlation based on *org:group*, the number of response constraints with a support of at least 30% reduces from 371 to 229 (a reduction of 38.3%). Adding the correlation requirement *A.Producer code = B.Producer code* results in a reduction from 371 to 100 response constraints.

Table 5. Pruning of constraints using correlations. The number of constraints reported are without filtering transitive reductions.

constraint	correlation	#constraints			
		supp=30	supp=35	supp=40	supp=45
response(A,B)	$\langle\langle$ *no correlation* $\rangle\rangle$	371	286	225	125
	A.org:group = B.org:group	229	180	163	114
	A.Producer code = B.Producer code	100	85	83	71
	|A.time:timestamp − B.time:timestamp| $\leq$ 4 days	226	172	139	112
precedence(A,B)	$\langle\langle$ *no correlation* $\rangle\rangle$	458	403	352	261
	A.org:group = B.org:group	274	249	240	237
	A.Producer code = B.Producer code	113	106	104	104
	|A.time:timestamp − B.time:timestamp| $\leq$ 4 days	325	281	274	217

We further analyzed the log for discriminatory patterns that may exist between fulfillments and violations of some constraints. Here, we present one such example of the *response(A,B)* constraint (where *A* corresponds to *First outpatient consultation* and *B* corresponds to *administrative fee - the first pol*). The event log contains 517 (non-ambiguous) fulfillments and 60 violations of this constraint. We considered the event attributes of *First outpatient consultation*, correlations involving these attributes, and the case attributes pertaining to the traces involving these activations.

The event log contains several case level attributes such as diagnosis and treatment codes (the reader is referred to [8] for a detailed description on these case attributes). We have grouped different variants of similar case attributes into a single attribute (e.g., the 16 diagnosis code attribute values are captured as a set of values under a single attribute). We have transformed the 577 activations into vector space using these attributes and their correlations and applied the J48 [23] decision tree learning algorithm. Out of the 60 non-conformant activations, we could find discriminant patterns covering 23 activations using these features. For example, five of the six activations whose value of *A.Section* is *Section 5* and *C.DiagnosisCodeSet* is $\{106, 823\}$ are non-conformant, i.e., TP=5 and FP=1 (*C* signifies a case-level attribute). Similarly, three of the four activations whose value of *A.Section* is not equal to *Section 5* and *A.Producercode* is *SGSX* are non-conformant, i.e., TP=3 and FP=1.

Fig. 4 depicts the annotation of a Declare map with performance information using the Extend Declare Map with Time Information plug-in. The map is color coded to easily pin-point bottlenecks based on flow times and the plug-in allows for the interactive exploration of a wealth of diagnostic information e.g., #activations, #fulfillments, etc. on the constraints.

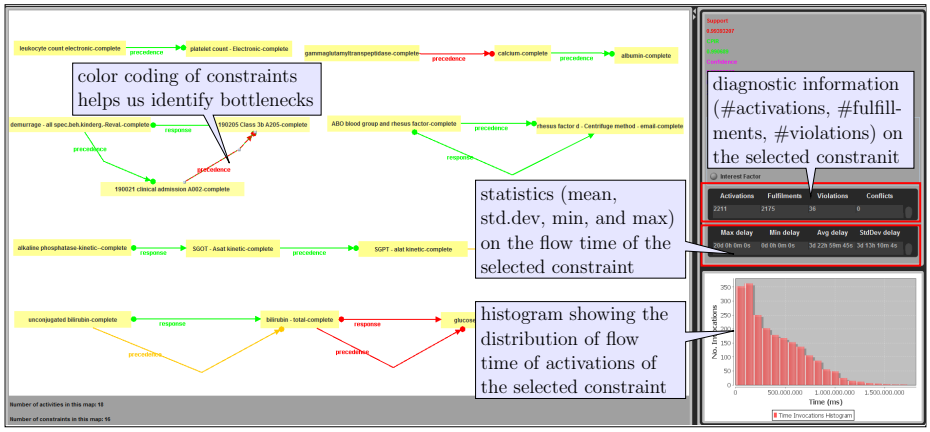


Fig. 4. Interactive visualization of a Declare map with diagnostic information such as flow time and bottlenecks.

6 Related Work

Techniques for the automated discovery of Declare maps from event logs have been proposed in [17–19]. In [19], the authors rely on a combinatorial exploration of all feasible constraint activations and evaluate the constraints' goodness based on their activation ratio, i.e., the percentage of traces where a constraint is activated. In [17], the authors adopt the classic *apriori* algorithm [5] to find frequent item sets in data mining to identify significant activations and propose several metrics such as support and confidence to prune uninteresting constraints. In [18], the authors extend the work of [17] through the incorporation of domain knowledge and techniques for pruning redundant constraints to uncover interesting constraints. None of these approaches exploit the availability of rich information in the form of data attributes in event logs.

Correlation is a critical topic when applying process mining techniques to event data recorded by non-BPM/WFM systems. Indeed, these systems just generate a list of events without providing a properly structured event log. To generate an event log, it is necessary to correlate events into process instances. In [21], the authors present an event correlation algorithm to discover the best correlation conditions from a set of candidates. This algorithm is used to generate an event log from a set of uncorrelated events. A probabilistic approach based on the Expectation-Maximization (EM) principle has been proposed in [10] for correlating events from unlabeled event logs (where case ids are not available).

In [20], the authors identify different types of correlations and investigate the problem of discovering event correlation from data recorded by service oriented systems. The authors also introduce the concept of *process view* to represent the process resulting from a certain way of event correlation. They argue that correlation is subjective and that multiple views are possible. For example, in a process for customer order handling, in one view, orders can be considered from the viewpoint of order lines and, in another view, the same orders can be considered from the viewpoint of deliveries. A collection of process views is called the *process space*.

Rozsnyai et al [24] propose approaches for automatically deriving correlations from arbitrary sources of data. An interesting part of their work is the automatic identification of attributes that might be correlated based on properties such as their type, cardinality and the domain of values. In this paper, we used the heuristic of considering attributes of similar type as comparable. It would be interesting to explore the applicability of the concepts proposed in [24] for correlating events for Declare map discovery.

7 Conclusions and Future Work

Declarative process maps discovered from event logs without any consideration for event and case attributes tend to result in inaccurate and incomprehensible results. In this paper, we exploited the data present in event logs to discover process maps only showing relevant and accurate constraints. We proposed a means of discovering significant correlations that exist between events and use these correlations to *prune constraints*, to *disambiguate event associations*, and to provide additional *diagnostic information*. Our evaluation using real-life logs demonstrates that the proposed approach is very promising, e.g., we are able to disambiguate up to 96.45% of events in a hospital log. In this paper, we focused only on positive relation constraints involving two activities. In the future, we would like to extend this to also cover constraints involving multiple activities and negative relations (e.g., not co-existence and not succession in Table 1). Also, the proposed approach relies on some heuristics such as the use of $\mu \pm \sigma$ of time difference for temporal correlations and the correlations of *like* attribute types. As future work, we would like to study the trade-off between completeness and efficiency of mining. Furthermore, we would like to evaluate our approach using more case studies.

References

1. XES Standard Definition (2009), `www.xes-standard.org`
2. 3TU Data Center: BPI Challenge 2011 Event Log (2011), doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
3. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
4. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative Workflows: Balancing Between Flexibility and Support. Computer Science - R&D pp. 99–113 (2009)
5. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: VLDB. pp. 487–499 (1994)

6. Barros, A., Decker, G., Dumas, M., Weber, F.: Correlation Patterns in Service-Oriented Architectures. In: FASE. LNCS, vol. 4422, pp. 245–259. Springer, Berlin (2007)
7. Binder, M., Dorda, W., Duftschmid, G., Dunkl, R., Fröschl, K.A., Gall, W., Grossmann, W., Harmankaya, K., Hronsky, M., Rinderle-Ma, S., Rinner, C., Weber, S.: On Analyzing Process Compliance in Skin Cancer Treatment: An Experience Report from the Evidence-Based Medical Compliance Cluster (EBMC2). In: CAiSE. LNCS, vol. 7328, pp. 398–413 (2012)
8. Bose, R.P.J.C., van der Aalst, W.M.P.: Analysis of Patient Treatment Procedures: The BPI Challenge Case Study. Technical Report BPM-11-18, BPMCenter.org (2011)
9. Burattin, A., Maggi, F.M., van der Aalst, W.M.P., Sperduti, A.: Techniques for a Posteriori Analysis of Declarative Processes. In: EDOC. pp. 41–50 (2012)
10. Ferreira, D.R., Gillblad, D.: Discovering Process Models from Unlabelled Event Logs. In: BPM. LNCS, vol. 5701, pp. 143–158. Springer, Berlin (2009)
11. IEEE Task Force on Process Mining: Process Mining Manifesto. In: BPM 2011 Workshops. LNBIP, vol. 99, pp. 169–194. Springer, Berlin (2011)
12. Kupferman, O., Vardi, M.Y.: Vacuity Detection in Temporal Model Checking. International Journal on Software Tools for Technology Transfer pp. 224–233 (2003)
13. de Leoni, M., Maggi, F.M., van der Aalst, W.M.P.: Aligning Event Logs and Declarative Process Models for Conformance Checking. In: BPM. LNCS, vol. 7841, pp. 82–97. Springer (2012)
14. Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. In: KDD. pp. 80–86. The AAAI Press (1998)
15. Ly, L.T., Indiono, C., Mangler, J., Rinderle-Ma, S.: Data Transformation and Semantic Log Purging for Process Mining. In: CAiSE. LNCS, vol. 7328, pp. 238–253 (2012)
16. Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring Business Process Compliance Using Compliance Rule Graphs. In: OTM Conferences (1). LNCS, vol. 7044, pp. 82–99 (2011)
17. Maggi, F.M., Bose, R.P.J.C., van der Aalst, W.M.P.: Efficient Discovery of Understandable Declarative Models from Event Logs. In: CAiSE. LNCS, vol. 7328, pp. 270–285. Springer, Berlin (2012)
18. Maggi, F.M., Bose, R.P.J.C., van der Aalst, W.M.P.: A Knowledge-Based Integrated Approach for Discovering and Repairing Declare Maps. In: CAiSE (2013), to appear
19. Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-Guided Discovery of Declarative Process Models. In: IEEE Symposium on Computational Intelligence and Data Mining. vol. 2725, pp. 192–199. IEEE Computer Society (2011)
20. Motahari-Nezhad, H.R., Saint-Paul, R., Casati, F., Benatallah, B.: Event Correlation for Process Discovery from Web Service Interaction Logs. The VLDB Journal 20(3), 417–444 (2011)
21. Perez-Castillo, R., Weber, B., Guzmn, I.R., Piattini, M., Pinggera, J.: Assessing Event Correlation in Non-Process-Aware Information Systems. Software & Systems Modeling pp. 1–23 (2012)
22. Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.A.: Imperative Versus Declarative Process Modeling Languages: An Empirical Investigation. In: BPM Workshops. LNBIP, vol. 99, pp. 383–394 (2011)
23. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
24. Rozsnyai, S., Slominski, A., Lakshmanan, G.T.: Discovering Event Correlation Rules for Semi-structured Business Processes. In: DEBS. pp. 75–86 (2011)
25. Schulte, S., Schuller, D., Steinmetz, R., Abels, S.: Plug-and-Play Virtual Factories. IEEE Internet Computing 16(5), 78–82 (2012)