A Formal Model for Process Context Learning

Johny Ghattas[1], Pnina Soffer[1], Mor Peleg[1,2]

[1]Department of Management Information Systems, University of Haifa, Israel, 31905
[2]Center of Biomedical Informatics, Stanford University, Stanford, CA, 94305
GhattasJohny@gmail.com, morpeleg, pnina @{mis.hevra.haifa.ac.il}

Abstract. Process models are considered to be a major asset in modern business organizations. They are expected to apply to all the possible business contexts in which the process may be executed, however not all of these are known a priori. Instead of identifying all contexts before the process is established, we propose to learn from runtime experience which contextual properties should be taken into account by the process model. We propose a model and an associated procedure for identifying and learning the relevant context categories of a process out of runtime experience. We postulate that the context of a process, namely, properties of the specific business case and environmental events, affects its execution and outcomes. However, when a process is launched, the exact effect and affecting variables are not necessarily known. Our approach aims at categorizing possible environmental conditions and case properties into context categories which are meaningful for the process execution. This is achieved by a context learning framework, presented in the paper.

Keywords: Business process context, Business process learning, Process goals, Soft-goals, Process model adaptation, Flexibility.

1 Introduction

Modern Organizations require their business processes (BP's) to be standardized and, at the same time, to be able to handle the variability of their environment. This variability relates to differing properties of cases handled by the process and to the unanticipated and changing requirements of the market and customers. A general term, addressing both the events and conditions in the environment and the specific properties of cases handled by the process, is the context of the process [1, 2]. Consider, for example, a customer care center, through which an organization captures its customer claims and follows up on them. Here, we consider whether the customer has a valid warranty or not to be a contextual property of the specific case. It is quite expected that the business case will be treated differently, depending on this variable.

Clearly, different contextual conditions may require different paths for the process to achieve its goals. To facilitate this, three main challenges need to be met. First, normally there is no obvious way to establish a full repository of all possible context variations that are yet to appear. Second, while it is possible to have information about an (almost) unlimited amount of case properties, we should be able to identify which specific properties have an effect on the process. Third, organizations need to know how to select their process paths per each one of these situations in order to achieve the best outcome.

In this paper we target the second challenge, focusing on developing a methodology for automatic learning of process context groups. Context groups cluster together process instances that have similar contexts, thereby limiting the number of context variations to be dealt with. This can be a first step towards defining process paths for each context group, such that taking that path would lead to desired process outcomes. For this purpose, we target an active process, namely, a process which has already been executed for a while, and acquired past execution data. Our basic assumption is that in these past executions, some cases were addressed "properly" according to their relevant contextual properties (although a relation between context and path selection was not necessarily formally specified). Other cases were not properly addressed, and this should be reflected in the performance achieved by the process for these cases, which should be lower when compared to the properly addressed cases. Hence, the proposed methodology is based on clustering process instance data of past executions, relating to their context, path, and outcomes. The starting point is when all this information is available, but it is not known which contextual properties are the ones that should be considered for path selection. Clustering based on the path and outcomes of process instance data finds the relevant groups of context, where each group is considered similar in terms of its process path and outcomes. Our vision is that once context groups are formed, a new process instance that has not yet been activated could be matched to an existing context group, in order to suggest a path that would yield the desired outcome.

The remainder of the paper is structured as follows. Section 2 presents our conceptual model for BP context, which is an extension of the Generic Process Model (GPM) [3, 4]. In Section 3, we provide our algorithm's architecture, illustrating each step through an example order provisioning process from the cellular service domain. Section 4 provides a review of related work and Section 5 summarizes this work and outlooks to future research.

2 The Conceptual Model for Business Process Context

We will first establish a formal definition of the generic concepts for a context learning framework. Our proposed model builds upon GPM and extends it to incorporate the relevant concepts for modeling process context.

2.1 The Generic Process Model (GPM)

GPM [3, 4] is a state-based view of a process including the concept of goals. Briefly, GPM offers a process model defined over a domain as a tuple < L, I, G>, as described below. Consider the state of the domain as the values of all its properties (or state variables) at a moment in time, the law L specifies possible state transitions as a mapping between subsets of states; I is a subset of unstable states, which are the initial states of the process after a triggering external event has occurred; G is a subset of stable states on which the process terminates, termed the goal of the process. Following this, a specific path taken by a process is a sequence of states, transforming by law or as a result of external events, until a stable state is reached. If the process model is valid, this stable state is in the goal set.

The process goal as addressed by GPM is a state meeting the conditions that should be achieved by the process. GPM distinguishes process goals from soft-goals, which are defined as an order relation on goal states [4]. In other words, soft-goals relate to the desirability of possible states in the goal set (all meeting the condition that terminates the process) according to defined business objectives. For example, the goal of a process may be a state where some treatment has been given to a patient, but a state where the treatment does not incur side effects is considered as “better” than a state where side effects are observed. Finally, GPM entails criteria for assessing the validity of a process, namely, its ability to achieve its goal [3]. It enables the analysis of a process model to identify causes for invalidity.

2.2 The Conceptual Model for Context Learning

Although GPM does not provide a context model, it can be extended to support such concepts. In this section we provide a detailed description of our GPM extension for context modeling.

We postulate that a process representation depends on a context if a process instance cannot be executed correctly without having some additional inputs regarding the values of case properties (e.g., whether the customer has a valid warranty) or events arising from the external environment (e.g., the customer changes his order). We extrapolate this definition to BP context, where the context of a process or a plan would be the set of all inputs provided to the process during its enactment lifetime.
In GPM terms, the external environment provides inputs to the process through the initial triggering of the process and through external events which reach the process during runtime. We denote the set of external events reaching the process during its execution as X and the set of case state-variable values known at the initial triggering of the process as I.

Note that our interest, while trying to learn process contexts, is to group in a meaningful way all possible context instances, that is, all possible <I, X> combinations, so these groups would represent specific business cases. As an example, service providers may implement different products, price plans and processes for supporting business customers and for supporting private customers. Hence, there are two major context groups - the corporate customer context group, which includes all context instances of business customers, and the retail customer context group, which includes all context instances of private customers. Each of these groups may be further divided in sub-groups to enable more fine-grained tailoring of paths for each sub group. For example, private customers may be divided by age, by bundles of services (cellular, data, content), etc. However, we should not reach a level of granularity where we have context groups that are too specific, to avoid over fitting the context groups with the specific context instances from which the context groups were identified.

Following this intuitive discussion, we first extend GPM by formalizing the concepts of process instance and of the context of process instances. Later on we define the concepts of behavioral and context-based similarity of process instances.

Definition 1(process context): a business process context C=<I, X> is the set of all inputs provided by the external environment to the process, where I marks the state variable values at the initial state, set at process instance triggering time; X is a set of external events, which affect the process instance at runtime.

The context of a specific process instance PI_i is obtained by assigning values to I and X of PI_i:

$$C_i = < I_i, X_i > .$$

For example, considering the case of a customer ordering a cellular phone and services, I would be the set of data the customer provides at order request time (characteristics of the phone he would like – slider phone, Bluetooth capabilities, camera, etc.; the services he wants – voice calls, SMS, internet connection; customer budget limits, etc.). X would be changes the customer introduced to his order some time after the order was initiated, e.g., upgrading into a more advanced package with more features. The context model for this example would be:

$C = < I, X >$, where:

```
I= {{Customer characteristics set}, {Phone characteristics
set}, {Set of Services required}, {Customer budget}}.
X= {{Order change during execution # 1}}.
```

Note that the effect of an external event over the process may be different depending on its arrival time at the process. For example, cancelling an order may affect the process differently if it it occurs before or after the handset is delivered. In the first case, the change would simply imply roll-backing the order in the system and returning the equipment to the logistical department, while in the second, the customer would be required to return the equipment before the order change.

Definition 2 (process instance): Given the context $C_i = <I_i, X_i>$, the execution of a *BP instance* would lead to the generation of a path P_i, which terminates either in a goal state $t_i \in G$, or in an exception state $t_i \in E$, where E is a set of stable states not in the goal ($E \cap G = \varnothing$).

Based on this, we can model a process instance (PI_i) by as $PI_i = <C_i, P_i, t_i>$, where $t_i \in G$ or $t_i \in E$, and P_i is a sequence of states, following GPM's definition.

The path (P_i) and the termination state (t_i) of a process instance (PI_i) constitute its behavior. In a perfect world, process instances that have similar contexts would follow similar paths to lead to a given termination state. However, our knowledge of the process context is partial. Under partial knowledge, we may not be aware of contextual variables whose different values may differently affect the process behavior, and can be considered "different contexts". Lacking such knowledge, we may group PIs that partially share the same context but exhibit different behaviors. This would not be an effective strategy for learning the best paths that for a given context would achieve desirable outcomes. Hence, process instances can be grouped considering two types of similarities:

Formatted: Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.4 cm + Indent at: 1.04 cm

(1) Contextual property-based similarity.

(2) Behavioral similarity.

Clearly, these two groupings are expected to be different, since not all contextual properties necessarily affect process behavior, and some properties may have a similar effect. Our interest is to identify a third type of grouping, *context groups definition*, namely, groups of instances whose contextual property-based similarity can predict some behavioral similarity.

In the following, we first discuss behavioral similarity identification. We continue by discussing contextual property-based similarity, and then rely on these two to develop criteria for context group definition.

Behavioral Similarity of Process Instances

Our objective is to group process instances that follow similar paths and result in similar termination states (goal or exception) into homogeneous groups. In order to establish behavioral similarity of process instances, we need similarity criteria for both termination states and paths of process instances.

Definition 3 (state similarity): Let s_1 and s_2 be two states, $s_i=(x_{i1}, x_{i2},\ldots x_{in})$, i=1, 2, where x_{ij} are state variable values. For a given similarity threshold ST, s_1 is *similar* to s_2 iff a distance measure D satisfies $D(s_1, s_2) \leq ST$. Notation: $s_1 \sim s_2$.

Note that technically, state similarity can be established by applying various clustering algorithms to state data. However, conceptually, similar states could be viewed as identical at some granularity level. For example, there may be many different states where a product has passed quality tests (with different values of test results). At a certain granularity level, all these states are identical (successfully passed). Similarity of paths and termination states of process instances is derived from Definition 3.

The derivation follows since termination is a specific state, and a path, according to GPM, is a sequence of states. Note, to this end, we neglected the ordering of states in a path, taking account only of the state variable values. Also note that hereafter we assume a given similarity threshold, so the existence of similarity can be established.

Definition 4 (Process instance behavioral similarity): Consider two process instances PI_i and PI_j, $i \neq j$, where: $PI_i = <C_i, P_i, t_i>$, $PI_j = < C_j, P_j, t_j >$. These process instances are considered *behaviorally similar* if and only if their path state variable values are similar and their termination states (either in the goal or in the exception set) are similar:

$$PI_i \sim PI_j \Leftrightarrow P_i \sim P_j \text{ and } t_i \sim t_j .$$

Grouping process instances by behavioral similarity yields clusters of similar instances as defined below.

Definition 5 (process instance cluster): A *process instance cluster* (PIC_k) is the set of all process instances PI_i which are behaviorally similar to each other:

$$PIC_k = \{PI_i, PI_j \mid PI_i \sim PI_j , i \neq j\} \quad \forall k .$$

Definition 5 implies that each process instance PI_i can be assigned to one specific process instance cluster PIC_k. Hence we can say that if a PI is assigned to one PIC it is not assigned to another PIC:

$$\text{Given i, k: } PI_i \in PIC_k \Rightarrow \forall l \neq k, PI_i \notin PIC_l .$$

We will discuss our technical approach for creating the PICs in Section 3. Basically, grouping instances into PICs as specified in Definition 5 is completely based on the behavior observed in actual process instances, that is, process path and termination data. It does not relate to the contextual properties, neither does it establish similarity of the contexts themselves, which we discuss next.

Contextual Property-Based Similarity

We now turn to discuss similarity of process instance contextual properties. Our analysis begins when we have information about the context of every process instance. However, we do not know which of these properties affect process behavior and how.

Definition 6 (contextual property): A *contextual property*, CP, is a logical predicate established over a subset of context variables, that is, state variables whose values are defined in *I* and *X*.

As an example, in the case of a service provider, the predicate "`customer_type = "Business customer`" is a contextual property as it establishes a logical predicate over the state variable `customer_type`, whose value is defined for states in the set *I*.

Definition 7 (contextual property-based similarity): Two process instances $PI_i = <C_i, P_i, t_i>$ and $PI_j = < C_j, P_j, t_j >$ are *contextual property-based similar* if $\exists$ contextual property CP such that $CP(C_i) = CP(C_j)$.

Note that, as opposed to behavioral similarity-based grouping, where a process instance can be included in one PIC only, here there might be a large number of groupings, each based on a different contextual property. A process instance can thus be included in more than one contextual property-based similarity group.

Context Groups

Above we showed how to identify behavioral similarity and contextual property-based similarity of PIs. Yet, we would like to find meaningful process instance groups, similar in their contextual properties, such that for each group, following a certain path would enable predicting its outcome. We term these groups *context groups*.

Definition 8 (Context group): A *Context Group* CG is a set of process instances such that: $CG=\{PI_i, PI_j \mid \exists CP_k: \forall i, j, (CP_k(PI_i)=CP_k(PI_j) \wedge P_i \sim P_j) \Rightarrow t_i \sim t_j\}$.

We assume that such context groups exist, and try to identify which contextual properties satisfy the implication relation of Definition 8. Two main difficulties need to be overcome. First, process instances in a context group may follow different paths and achieve different termination states, thus they are not necessarily behaviorally similar. Second, the possible number of contextual properties increases exponentially with the number of contextual state variables. Note that in addition to these two difficulties, in real-life situations not all the contextual information is available. There might be state variables of which partial or even no information is available, or the actual data might be "noisy". The quality of the data may be manifested as statistical errors when similarities are assessed, accounted for by the procedure presented in Section 4. However, the rest of this section addresses complete and "clean" information.

In order to identify context groups, we analyze the consequences of Definition 8. As a result, we derive two postulates characterizing behavior of instances in a context group.

Consider two process instances, PI_i and PI_j, such as: $PI_i = <C_i, P_i, t_i>$ and $PI_j = < C_j, P_j, t_j >$, $i \neq j$, and assume we know the actual context groups. There are eight possible combinations of whether these

instances are (a) in the same context group CG_k, (b) similar at the path, and (c) similar at the termination state, as detailed in table 1. The last column of the table indicates what combinations can occur according to Definition 8; the implication of Definition 8 is false only if two PIs that are in the same context group and follow similar paths result in different termination states.

Table 1. Possible combinations of CG, path and termination state similarities. (T= True, F= False)

Case #	$PI_i, PI_j \in CG_k$	$P_i \sim P_j$	$t_i \sim t_j$	Can this combination occur?
1	F	F	F	T
2	F	F	T	T
3	F	T	F	T
4	F	T	T	T
5	T	F	F	T
6	T	F	T	T
7	T	T	F	F
8	T	T	T	T

Examining Table 1, we can see that for process instances not in the same context group, all combinations of paths and termination states are possible (cases 1-4 in Table 1). In addition, for instances in the same context group that have different paths (cases 5 and 6 in Table 1), similar or different termination states are possible. If the instances are in the same context group and have similar paths, their termination states should be similar (case 8), and cannot be different (case 7). However, Table 1 relates to known context groups, while in our problem context groups are unknown. Hence, the only conclusive assertion we can make with respect to two process instances is that they are not members of the same context group if their paths are similar and their termination states are not, as formalized in Postulate 1.

Postulate 1: Let {PI} be a set of process instances grouped by some contextual property CP_k $\{PI\}=\{PI_i, PI_j \mid \exists\ CP_k: \forall i, j, CP_k(PI_i)=CP_k(PI_j)\}$. If $\exists\ PI_i, PI_j \in \{PI\}$ such that $P_i \sim P_j$ and $\neg\ (t_i \sim t_j)$ then {PI} is not a context group.

To illustrate this, consider an ordering process of a cellular service provider. Assume that business customers may order a specific package and be given a specific offer for a price. Also assume that some will accept this offer and some will reject it. We may conclude that "business customers" is not a context group, and some finer grained contextual property should be used in order to divide business customers further into groups where the outcome (accept or reject) can be predicted based on the path (offer given).

Postulate 1 provides a conclusive criterion for excluding a set of contextually similar process instances as a context group. However, we do not have a conclusive criterion for positively asserting a set of PIs as a context group. We cannot rely on case 8 in Table 1 for forming such criterion because similar paths and similar termination states are also possible for two process instances which are not in the same context group (case 4 in Table 1). Nevertheless, we may assume that groups of contextually similar process instances (groups from cases 5 through 8) form a context group if they consistently (for each path from a set of different paths) achieve similar termination states given a similar path (only case 8 satisfies this condition). In the example above, consider two sets of contextual property-based similar process instances: group1 including business customers whose number of handsets is below 10 and group 2 of business customers whose number of handsets is between 10 and 20. Assume that both groups consistently accept one package and reject a second package. We may consider these two groups as one context group, since the differences in their contextual properties are not manifested in their behavior (they have the same behavior for a set of paths).

We use Figure 1 to explain the principles for proposing context groups based on consistent behavior similarity. As shown in Figure 1, PICs (columns) include PIs that follow the same path and yield the same termination state. However, a PIC can contain PIs belonging to different context groups (CGs). For example, PIC 2 contains a set of process instances that can be partitioned based on some contextual property into four subsets $\{PI\}_1$ through $\{PI\}_4$. The entire set of PIs contained in PIC2 cannot form a CG because although all PIs that are contained in PIC2 terminate in X when they follow Path B, when PIs contained in $\{PI\}_1$ and $\{PI\}_3$ follow Path A they terminate in X but when PIs contained in $\{PI\}_2$ follow Path A they terminate in Y. Therefore, PIs contained in $\{PI\}_2$ cannot belong to the same CG as PIs contained in $\{PI\}_1$ or $\{PI\}_3$. We can propose CGs based on similarity of behavior. In Figure 1, $\{PI\}_1$ and $\{PI\}_3$ have instances in exactly the same PICs, hence their behavior is consistent (they terminate in X when they follow paths A, B, or C) so they can be considered one context group. As explained above, $\{PI\}_2$ does not belong to the same context group as $\{PI\}_1$ or $\{PI\}_3$. $\{PI\}_4$ does not violate Postulate 1, but it has no instances in PIC 1 or PIC4, hence it is not consistent in its behavior with the other sets of PIs and is therefore not considered in the same context group with

them. In summary, based on similarity of behavior we can see three CGs: $\{PI\}_1 \cup \{PI\}_3$; $\{PI\}_2$; and$\{PI\}_4$. The CGs are formed by splitting each PIC into subsets of instances based on context similarity and combining subsets that exhibit similar behavior across all PICs into CGs.

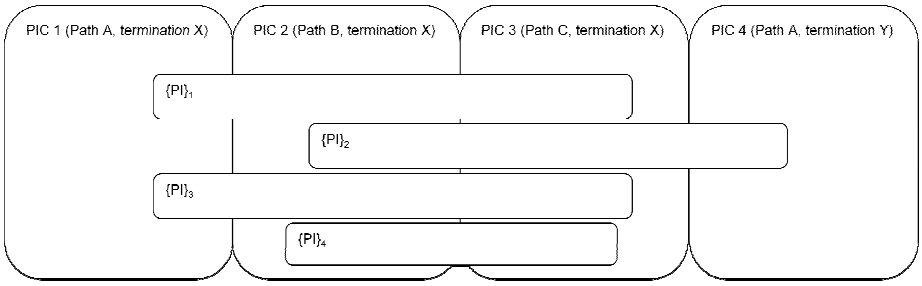

Figure 1. PICs for a combination of path and termination states.

In general, given two sets of contextual property-based similar process instances such that each one complies with Postulate 1; if they consistently exhibit similar behaviors we can relate to them as being in the same context group. This is formalized in Postulate 2.

Postulate 2: Let $\{PI\}_1$ and $\{PI\}_2$ be sets of process instances that are contextual property-based similar, such that each one complies with the criterion of Postulate 1. If $\forall PIC_k$, $\{PI\}_1 \cap PIC_k \neq \emptyset \Rightarrow \{PI\}_2 \cap PIC_k \neq \emptyset$ and $\{PI\}_2 \cap PIC_k \neq \emptyset \Rightarrow \{PI\}_1 \cap PIC_k \neq \emptyset$, then $\{PI\}_1 \cup \{PI\}_2$ is considered a context group.

Postulate 2 enables us to join small context groups into larger ones if they consistently exhibit similar behavior, thus reducing the number of context variants to be addressed.

3 An Approach for Learning Context Groups of Business Processes

Based on the model presented in Section 2, we have established a procedure which implements the context group identification in a five stage algorithm, as schematized in Figure 2.

Our starting point is a database of process instances, which includes the following data:

Formatted: Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0 cm + Indent at: 0.63 cm

(1) Path data, organized as a set of states, where each state is a vector of state variable values.
(2) The termination state, which is provided as a vector of state variable values.
(3) Context data, which is composed of the initial state (*I*), in the form of a state variables vector, and a set of state variable vectors representing the external events received during the process execution.

The basic principles of the procedure are as follows. Behavioral similarity of process instances is identified using a clustering algorithm, where process instances are assigned to a process instance cluster ID. Termination state similarity assessment is achieved based on predefined rules, as the termination state modeling is assumed to be part of the BP model, and hence we have the termination states categorized a-priori. Note that process instance clusters (PICs) relate to both path and termination state similarity. Considering the contextual properties, we use machine learning techniques to find combinations of contextual properties that best predict the behavioral similarity category (namely, the PIC) of each process instance. The result is a partition of the instances to sets which are both contextually and behaviorally similar.

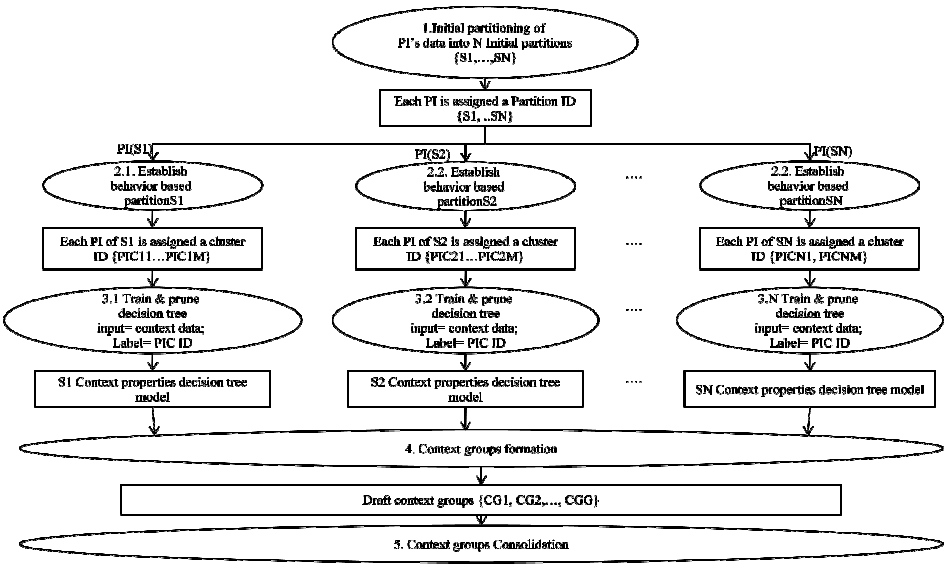

Figure 2. Architecture of the context groups learning algorithm. PI- process instance; S- Initial partition

Finally, based on Postulate 1, we exclude the sets that violate this condition, and based on Postulate 2, we join groups whose behavior is similar. The steps of the algorithm are as follows:

Step 1: Partition the process instances into N partitions based on existing domain knowledge (see Figure 2). A-priori knowledge comes from domain experts as well as from the literature. The objective of this step is to identify groups of business cases that are relatively uniform in their characteristics.

For each partition (S_i), we separately apply the following three steps of the algorithm.

Step 2: Establish the behavioral similarity of the process instances PI_i (step 2 in Figure 2). This is done in three steps:

Formatted: Numbered + Level: 1 + Numbering Style: a, b, c, … + Start at: 1 + Alignment: Left + Aligned at: 0 cm + Indent at: 0.63 cm

(a) Establish the process instance path (P_i) similarity using a clustering algorithm which is applied to the set of PI state vectors. The objective is to assign to each process instance a process path similarity category. This step results in the identification of p groups of instances, assumed to be similar at the process path level. The number of path similar clusters generated, p, would be selected according to goodness of fit criteria, such as Akaike Information criteria (AIC) [5]. The clustering algorithm can be applied several times, achieving a series of clustering results with an increasing number of clusters for each clustering set. Finally, the best cluster set is selected as the one that attains the first minima of the ratio of AIC changes.

(b) Establish the termination state (t_i) similarity. The termination categorization is based on a set of predefined rules stated as basic logical predicates over the termination state variables. We assume having t different categories of termination state groups.

(c) Establish behavioral similarity by selecting the process instance categories which are simultaneously similar at process instance path and termination levels. This is done by splitting up the clusters formed in part (a) into groups in which the PIs have similar termination states. Out of these categories, select only the categories which present a significant number of instances, which we consider as categories including more than 5% of the total number of instances. The rest of the categories are ignored. Now we should have each PI tagged with a process instance behavior cluster (PIC), with PICs ranging from 1 up to M.

Step 3: Establish the process contextual properties (CPs). This is accomplished by training a decision tree algorithm, using the context data as inputs and the PIC IDs as dependent variable (label). The objective of using the decision tree is to discover the semantics behind each PIC. Based on the context data of the PIs clustered in each PIC, we use a modified Chi-squared Automatic Interaction Detection (CHAID) growing decision tree algorithm [6] to construct the decision tree that represents the context groups and their relationships. We provide CHAID with the context data of the PIs and with the PIC ID of each PI, which was deduced in step 2 according to the path and termination state data of the PIs. The PIC ID serves as the dependent label. CHAID tries to split the context part of the PI data into nodes that contain PIs that have the same value of the dependent variable (i.e., which were labeled in step 2 by the same PIC ID). Each decision tree path from the source node to each one of its leaf nodes represents a different contextual property (CP – a predicate in the contextual state variables). Each leaf node contains a certain distribution of instances of each PIC, allowing the identification of the most

probable PIC for that leaf. We use the Chi-Square criteria for tree growing and pruning. The tree is cross-validated using a k-fold cross-validation strategy (k=15).

Step 4: Form the context groups (CGs). Based on Postulate 1, for all tree paths, eliminate a path if it contains instances from PICs that have similar paths but different termination states[1]. We also eliminate nodes which include similar levels of different PIC's and hence have no clear output. The remaining paths are assumed to be context groups.

Step 5: Join (consolidate) context groups if their instances are included in the same PICs, based on Postulate 2.

In order to illustrate the proposed algorithm, consider a simple example of the ordering process of a cellular service provider. The process deals with ordering cellular handsets and services by both business and private customers. The context of the process consists of the customers' details (e.g. age, location, whether this is an existing customer or a new one, average expenses per month, major usage (voice, data, mobile modem connection, content services, etc.)), and the details of the services the customer requests (voice, SMS, data mobile modem, content services, mobile TV, etc.). We assume that the service provider offers service packages that include voice, data, and content services, based upon a second generation (2G) and third generation (3G) network technology. We also assume the possible offering of Blackberry handsets and services.

The first step (Step 1) would consist of identifying the initial partitions, e.g., S_1= "Business customers" and S_2 = "Private customers". This partition relies on domain knowledge, identifying these two categories as different lines of business, very different in their business goals and scenarios, and served by different organizational units.

We proceed to focus on each category separately and apply the next four steps of the algorithm to each category. Considering, for instance, S_1, we would take all PIs included in this partition and apply to their path data the clustering algorithm. Suppose that it would result in three path-similar categories: category 1 including 45 % of the instances, category 2 including 30% of the instances, and category 3 containing the remaining 25 % of the instances.

Next, we categorize the termination properties of these process instances, based on the following two rules provided by the business expert:

```
Rule 1: Normal termination: Customer_feedback= "Confirmed" AND
Package_status = "delivered" .

Rule 2: Customer reject: Customer_feedback = "cancelled order"
AND Package_status = "returned" .
```

Now we have three path categories and two termination categories, hence we have six potential PICs that are the different combinations of these two sets. However, assume that not all combinations exist in the process instances in the database. The PICs that represent the existing combinations are presented in Table 2.

Table 2. PICs found in the example.

PIC #	Path similarity category	Termination state rule
1	1	1
2	2	1
3	3	1
4	1	2

We assume that all these combinations have more than 5 % instances out of the total sample and hence all of them would be considered as relevant PICs.

Step 3: We now proceed to the identification of the contextual properties of these four PICs.

We use the decision tree algorithm which we train with the PIC IDs and which results in the decision tree schematized in Figure 3:

[1] As we need to account for a certain level of error, we consider that if a leaf node contains l_1 instances of PIC_i and l_2 instances of PIC_j, with $l_1 > l_2$, where PIC_i and PIC_j are similar at path level and different at termination state level, we would eliminate the path leading to this leaf node if $l_2/l_1 > 10$ %.

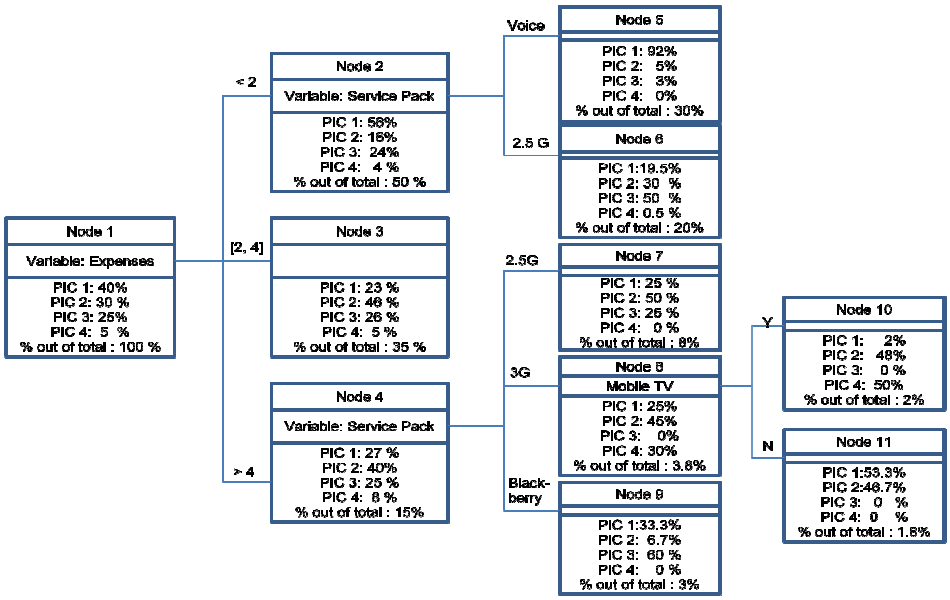

Figure 3. Hypothetic decision tree for a service provider ordering process. The tree partitions PIs according to contextual properties that best explain the similarities in path and terminal state.

The tree has seven paths, all starting with a partition over the state variable `expenses`, over the ranges of less than 2000 $/month, between 2000 and 4000 $/Month and more than 4000 $/month.

For the first range, the decision tree identified a sub-partition based on the `service_pack` variable, which divides the instances into the values of `Voice` and `2.5G` packs. For the third range it identifies a sub-partition based on the `service_pack` variable, dividing the instances into the values of `2.5G`, `3G` and `Blackberry`. For the second range of `expenses`, the decision tree did not identify any good sub-partition of the set of PIs. For `service_pack` = `3G` (node 8), an additional partition is proposed, using the state variable `mobile_TV`, resulting in nodes 10 and 11.

As a result, we have seven leaf nodes that stand for seven contextual properties. These should be checked for compliance with Postulate 1. According to Postulate 1, PICs whose paths are similar and termination state is not cannot be in the same context group. Examining Table 2, such PICs are PIC1 and PIC 4. Considering the decision tree results, leaf node 3 includes PIs belonging to both PIC1 and PIC4, hence, it cannot be considered as representing a context group[2]. **Table 3**~~**Table 3**~~ presents the contextual property predicates associated with each leaf node in the decision tree and an indication whether this CP stands for a context group.

Field Code Changed

As seen in Table 3, we have identified six context groups, associated with CPs 2, 3, 4, 5, 6 and 7.

Recall, our motivation for identifying context groups was the assumption that the properties defining these groups affect path selection and outcome (termination state) of process instances. Examining the details of the context groups we identified, note that the predicted PIC for group 6 (node 10 in the decision tree) is PIC4, whose termination state is defined as “customer reject” (an exception termination state). Hence, the PIs in this node are a valuable source of information about process exceptions. The indication is that 50% of the customers which fall into that context group (namely, expenses over 5000$ a month who ordered a 3G service pack and a mobile TV), when treated following path 1 have rejects concerning their orders. In contrast, 48% of this population was treated following path 2 (PIC2), which resulted in a successful termination. The remaining 2% were treated successfully following path 1, and they are considered as not characterizing the behavior of this context group. This information can be used for establishing decision rules that relate the selected path to the context group in future executions of the process.

[2] Note that leaf nodes 6 and 10 include both PICs too but the level of PIC4 in node 6 is very low relative to PIC1 (0.5%PIC4/19.5% PIC1 = 2%, does not exceed the 5% threshold), and vice versa for node 10. Hence, we consider these as being within an acceptable margin of error.

Table 3. Context properties based on the decision tree of Figure 3. The predicted PIC in the fourth column is calculated as the most probable PIC in each leaf node (provided in the second column).

CP#	Leaf Node	CP predicate	Predicted PIC	EAR><TITL sible PICs	Is it a CG?
1	3	2< expenses < 4	2	1, 3, 4	**No**
2	5	expenses < 2 *and* service_pack= Voice	1	2, 3	Yes
3	6	expenses < 2 *and* service_pack= 2.5G	3	1, 2	Yes
4	7	expenses > 5 *and* service_pack= 2.5G	2	1, 3	Yes
5	9	expenses > 5 *and* service_pack= Blackberry	3	1, 2	Yes
6	10	expenses > 5 *and* service_pack= 3G *and* mobile_TV = Y	4	2	Yes
7	11	expenses > 5 *and* service_pack= 3G *and* mobile_TV = N	1	2	Yes

Next, based on Postulate 2, we see that the groups corresponding to context properties CP3 and CP5 can be joined into a single CG as they both comply with Postulate 1, include instances belonging to PICs 1, 2 and 3, and have the same predicted PIC. Note that on the basis of Postulate 2 it would also be possible to join the groups corresponding to CP2 and CP4 into the same group. However, these have a different predicted PIC. Assuming some business logic driving the path selection in the organization (although not in a conclusive manner), we would leave them as separate groups. Following postulate 2, we decide not to join CP2 and CP7 into the same CG as they do not share PIC3 in their behaviors. Hence we remain with the following final context groups shown in Table 4.

Table 4. Final context group list.

CG#	Leaf Node	CP predicate corresponding to the CG	Predicted PIC	Other possible PICs
2	5	expenses < 2 *and* service_pack= Voice	1	2, 3
3	6+ 9	(expenses < 2 *and* service_pack= 2.5G) *or* (expenses > 5 *and* service_pack= Blackberry)	3	1, 2
4	7	expenses > 5 and service_pack= 2.5G	2	1, 3
6	10	expenses > 5 *and* service_pack= 3G *and* MobileTV = Y	4	2
7	11	expenses > 5 *and* service_pack= 3G *and* MobileTV = N	1	2

4 Related Work

Context awareness has just started receiving attention in the business process area. Examples include [2, 7-9], as well as our own previous work [1]. Through all these works, some agreement can be found about context as being the impact that the environment has over the process. However, to the best of our knowledge, the identification of relevant contexts for a given process has not been formally or algorithmically approached so far. Xia and Wei [9] define context as a set of business rules, each of which represents a small unit of knowledge of business management (business goal, customer requirements, business strategy, etc.). In our proposed procedure the context groups are represented through predicates, much in-line with their proposal. Yet, their definition is far from specifying how a context is formally established, leaving to the user to edit and define the context manually. Rosemann and Recker [8] propose a context-aware process design methodology in order to approach the problem of identifying contextual variables that drive the need for flexible business processes. Their definition of context remains intuitive and not formally stated as in our case. In addition, their methodology is qualitative in nature and can benefit from our formal model in order to rely less on the process designer's judgment. It can help the designer by focusing his effort on a smaller set of context groups rather than a whole set of process instances.

Context Awareness has been addressed in different domains, such as knowledge management [10] and context aware applications (e.g., mobile applications taking into account the location, the profile of the user and past usage) [11]. Additionally, context modeling and context reasoning has been largely investigated in artificial intelligence and in cognitive science. In these domains, several researchers [12-15] have pointed out that our knowledge of context is always partial; a context is primarily a subset of an individual global state, which is a partial and approximate theory of the world from some individual's perspective. As stated by Giunchiglia and Ghidini [13], context reasoning is local, that is, it is applicable only within the frame of a context and may be invalid for a different context (principle of locality). In addition, the set of facts that an individual takes into consideration in order to draw a

conclusion, via deductive reasoning, is a small subset of his/her entire knowledge. This relates to our model of identifying the relevant contextual properties for each set of behaviorally similar process instances. The context property is locally-relevant only for the set of the process instances it represents.

While reasoning, people can switch between context views, depending on the relevance of the context to the next task that they are trying to accomplish. However, while doing so, compatibility of the reasoning performed in different contexts should be maintained (principle of compatibility) [13]. In our model, different ways of defining termination states with respect to goals may result in different affecting contextual properties. Still, they should all be compatible and rely on the same set of data.

Buvac et al. [15] argue that there may be multiple representations at different levels of details per any specific context. There are three dimensions in which representation of context may vary: partiality (knowledge being partial), approximation, and perspective. The level of approximation can be used to define a partial order of contexts. The appropriate level of approximation depends, among other things, on the problem to be solved. The perspective determines the content of a particular context. The authors also propose that the relationships between context definitions should be a hierarchy, described using subsumption relationships. In addition, they claim that there is a single outermost context definition. We mainly address granularity of the definition of a CG which corresponds to perspective (e.g., postulate 2 provides the criteria that we use to define the perspective under which we consider that two groups can be joined into one larger group; using a finer-granularity of context would result in smaller CGs). To some extent, we also relate to partiality (allowed statistical error in the procedure).

Our approach differs from a case-based reasoning (CBR) approach [16][17], which uses a case-base of process instances. For a given process instance awaiting execution it proposes a similar process instance from the case-base. The main differences between the approaches are: (1) CBR systems do not establish the context of the process as a main building block to compare process instances. Instead, the problem definition is left to the end user with no formal methodology to establish it; (2) CBR approaches do not establish any compatibility criteria like postulates one and 2 in our model. We consider that our approach is complementary to CBR systems and may provide them with a formal and systematic way for defining problem similarities, as well as a systematic approach for considering the process outcomes while querying for similar cases.

5 Discussion and Conclusions

While it is commonly known that the context of a process should be taken into account during execution in order to achieve the desired outcome, little has been done so far for systematically supporting this. This paper proposed an algorithmic approach for identifying context groups out of past runtime experience. Relating to context groups allows us to reduce the analysis of process behavior to a set of groups instead of addressing each one of the process instances individually. Our approach is to deduce the context groups from both the behavioral similarity of process instances and their contextual characteristics. The paper presents a formalization of these similarities, which is the basis for a context learning procedure. The procedure allows us to automatically deduce the context groups which affect the process execution and outcomes. While the example given in this paper is hypothetical, we have experimented with the proposed procedure applying it to a process taken from the health-care domain (treating urinary tract infection patients). The data included 297 patient records collected in a hospital. The state vector of a process instance included 80 state variables. Applying the procedure, we identified five context groups with an accuracy measure of 92% for 59% of the considered instances. We are currently evaluating the clinical implications of these results [18].

Our proposed procedure, as presented in Section 3, is derived from the formal model presented in Section 2. Yet, there may be other procedures that can be derived from these principles. In particular, for different domains of knowledge, different clustering and learning algorithms may be used within the framework of the proposed procedure.

Our current approach has some limitations. First, as stated earlier, we do not know for certain which contextual properties are the ones that really affect behavior. Second, there is a possibility that some context-relevant variables are unknown and hence their data was not collected. As a consequence, our results are bounded to a certain level of error, which we cannot eliminate. We face a similar situation of bounded statistical error when establishing the similarity of process instance paths; clustering selects the set of features that best represents the data, while ignoring the rest. The similarity threshold discussed in Definition 3 is not defined using a semantic definition but is determined statistically by the clustering algorithm. The dependency between different variables within the path data may well affect the accuracy of the clustering results, although using robust feature selection algorithms prior to clustering reduces this significantly. Finally, termination states are currently roughly categorized as

being within the process goal or not. In future, we intend to allow for assessing the level of soft-goal attainment as part of the terminal state similarity definition.

Our context-learning algorithm is a first step towards learning the process paths that should best be adopted for each context group and adapting the process model schema accordingly and this is our main future research direction. Future research would also include the evaluation of our algorithm though case studies from different domains. Referring to the algorithm evaluation program, While statistical measures can evaluate the significance of the predictions made by the context groups, the practical consequences of the result still need evaluation. In the long term, the identified context groups should support an improved path selection and, as a result, an improvement in business performance measures of the process under consideration. However, this result can only be evaluated over time. In the short term, the identified context groups can be evaluated by domain experts However, while expert evaluation has the advantage of relying on specific domain knowledge, it remains subjective and dependent on current domain knowledge. In contrast, the objective of our algorithm is to discover context groups that have not been known a-priori. Hence, domain experts may indicate the extent to which the proposed context groups seem logical, but the real evaluation will be in the long term.

References

1. Ghattas J., Soffer P., Peleg M.: A Goal-based approach for business process learning. Workshop on Business Process Modeling, Development, and Support (BPMDS'08), in conjunction with CAISE'08; Montpellier, France. (2008).
2. Ploesser K., Peleg M., Soffer P., Rosemann M., Recker J.: Learning from Context to Improve Business Processes. BPtrends 2009(1):1-9. (2009).
3. Soffer P., Wand Y.: Goal-driven Analysis of Process Model Validity. Advanced Information Systems Engineering (CAiSE'04) (LNCS 3084); 2004. p. 521-535. (2004).
4. Soffer P., Wand Y.: On the Notion of Soft Goals in Business Process Modeling. Business Process Management Journal;11(6):663-679.(2005).
5. Akaike H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control 1974;19(6):716-723.(1974).
6. Kass G.V.: An Exploratory Technique for Investigating Large Quantities of Categorical Data. J of Applied Statistics ;29(2):119-127.(1980).
7. Ludget A.A., Heiko M.: Exploiting User and Process Context for Knowledge Management Systems. Workshop on User Modeling for Context-Aware Applications at the 8th Int. Conf. on User Modeling; Sonthofen, Germany. (2001).
8. Rosemann M, Recker J, Flender C, Ansell P.: Context-Awareness in Business Process Design. 17 th Australasian Conference on Information Systems; Adelaide, Australia. (2006).
9. Xia Y, Wei J.: Context-Driven Business Process Adaptation for Ad Hoc Changes. IEEE International Conference on E-Business Engineering; p. 53-60.(2008).
10. Raghu TS, Vinze A.: A business process context for Knowledge Management. Decision Support Systems ;43(3):1062-1079.(2007).
11. Mikalsen M., Kofod-Petersen A.: Representing and Reasoning about Context in a Mobile Environment. Schulz S, Roth-Berghofer T, editors. Proceedings of the First International Workshop on Modeling and Retrieval of Context. CEUR Workshop Proceedings; Ulm, Germany; p. 25-35. (2004).
12. Giunchiglia F.: Contextual reasoning. Epistemologia - Special Issue on "I Linguaggi e le Macchine" ;XVI:345-364.(1993).
13. Giunchiglia F, Ghidini C.: Local Model Semantics, or Contextual Reasoning = Locality + Compatibility. Artificial Intelligence ;127(2):221-259. (2001).
14. Benerecetti M., Bouquet P., Ghidini C.: Contextual Reasoning Distilled. Journal of Experimental and Theoretical Artificial Intelligence ;12(3):279-305.(2000).
15. Buvac S., Buvac V., Mason I.A.: Metamathematics of contexts. Fundam. Inform ;23(2/3/4):263–301.(1995).16.Aamodt A.: Case based reasoning: foundational issues, methodological variations and system approaches. AI Communications ;7(1):39-59.(1994).
17. Weber B., Rinderle S., Wild W., Reichert M.: CCBR-Driven Business Process Evolution. Proc. ICCBR'05; p. 610-24. (2005).
18. Ghattas, J., Peleg M., Soffer, P.: Learning the Context of a Clinical Process. Accepted for publication, 3d International Workshop on Process-oriented information systems in healthcare, BPMDS, (2009).