Diagnosing Business Processes Execution using Choreography Analysis

Diana Borrego, María Teresa Gómez-López, Rafael M. Gasca and Irene Barba

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Sevilla, Spain
{dianabn,maytegomez,gasca,irenebr}@us.es

Abstract. This work presents a proposal to diagnose business processes that form a global process using a choreography analysis. The diagnosis is based on distributed diagnosis since the business process is formed by a process orchestrations modelled by a set of activities. These business processes have two different types of activities, with internal and external interaction. In this paper the knowledge of the whole business process is divided in different processes. In means that each user has a local point of view of the information of the organization, it also happens in distributed system, where neither agent has global information of how the system is modelled. This work propose a methodology to diagnose the business processes, analyzing only the interactions between the activities of different processes. In order to perform the fault detection for business processes, an algorithm has been defined based on distributed diagnosis. Also some definitions about model-based diagnosis have been redefined to be adapted to business processes diagnosis.

Keywords: Choreography Analysis, Business Processes Diagnosis, Model-Based Diagnosis.

1 Introduction

Currently the business processes can be composed of different subprocess and a large number of activities that interact by means of a choreography with the same process or another.

Fault diagnosis permits to determine why a business process correctly designed does not work as it is expected. The diagnosis aim is to detect and to identify the reason of an unexpected behavior, or in other words, to identify the parts which fail in a process orchestration. Our proposal is based on DX community approaches [12], [9]. These works were proposed to find out the discrepancies between the observed and correct behavior of a system. In this paper we adapt the DX methodology to business processes fault detection. In the business processes, each user has a local point of view

The traditional diagnostic tools can be considered as a single diagnostic agent with a model of the whole system to be diagnosed, however, in some systems a single agent approach is not desirable. Moreover the integration of knowledge into one model of the system is infeasible if the system is too large, dynamic or distributed over different local entities. In some systems, the knowledge integration can proceed from different local diagnostic processes situated in different nodes (it is called spatially distributed) or from different fields of expertise (it is called semantically distributed [6]). When a business process is performed by different users, the full knowledge is unknown since is divided into different process, and the behavior of the process cannot be studied is a global way. Thereby, the business process diagnosis can be compared to the distributed diagnosis.

In previous works, the fault diagnosis for systems is classified as follows:

– **Centralized approach**. In general, traditional model-based diagnosis is centralized, and the diagnostic algorithm is run on an only system. This system captures all the

observations in order to perform the global diagnosis. Relating it to business process, it can be the diagnosis of a business process where all the activities that form the process are known.

- **Decentralized approach**. In order to obtain a global diagnosis of a system, a central coordination process and a local diagnoser for each subsystem that form the whole system are required. Some examples were presented in [4][10][7], where local diagnosers are communicated to a coordination process obtaining a global diagnosis. This type of knowledge distribution can be compared with the orchestration of a set of process to achieve an objective, where the central agent is in charge of the combination of the processes.
- **Distributed approach**. This type of systems represents the aim of this paper. It uses communication by a set of local diagnosers instead of requiring a global coordination process such as in a decentralized approach. In the bibliography, there are several proposals where there is no centralized control structure or coordination process [1][5][11][13][14]. Each local diagnoser is communicated directly with other diagnosers. In these systems the model is distributed, the diagnosis is locally generated and the inter-component consistency should be satisfied. Our solution is centered here, where each process that forms the global business process counts on a local agent which is in charge of the communication and the local diagnosis. This implies that the different process performs a part of the choreography to obtain the objective of the full process.

As it has been commented for each type of distribution of systems, this type of problematic is similar to business processes diagnosis, where the orchestration of the different processes and the different activities form an unknown process in a global way [15]. In the case of business processes, each business process does not know the rest of the business processes that take part in the orchestration to achieve a common objective. Therefore, in the solution given in this paper each business process counts on a local diagnoser to carry out the diagnosis tasks. These local diagnosers will communicate between them to carry out the diagnosis based on external interactions. An important difference of our solution with respect to previous ones is that it is not necessary to carry out a complete monitorization of the business processes to know what activities are failing, so that we do not need to perform the measurement of all the properties related to a running workflow instance. A previous solution to this problem [8] applies chronicles recognition to monitor and diagnose the behavior of software components.

The possible mistakes in a business process are: *(a)* the creation of a model that does not correspond to the real problem, *(b)* one or more than one activities are not executed as it was modelled, and *(c)* errors in the definition of the interactions between the activities of a process, or between the activities of different processes.

In order to develop the business process diagnosis, an architecture and an algorithm are proposed to obtain a precompiled structure that can be monitorizated depending on the external interactions and makes possible an improvement of the temporal efficiency, since this structure is prepared offline.

This work is organized as follows: Section 2 presents concepts related to centralized and distributed diagnosis adapted to business process diagnosis. Section 3 shows the description of the model of Business process related to distributed approach, and presents a motivating example to explain our proposal. Section 4 presents the distributed algorithm using the previous example. Finally, some conclusions and future work are presented.

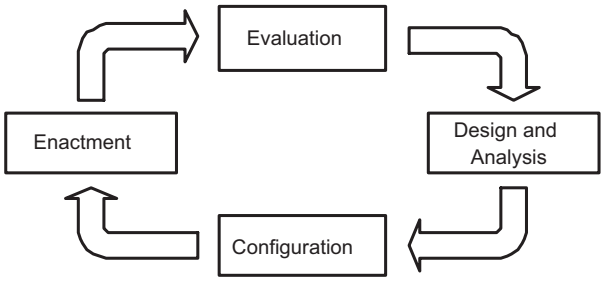

Fig. 1: Business process lifecycle.

2 Fault Diagnosis of Business Processes

Fault detection and identification of faulty activities of business processes are very important from the strategic point of view of the industries, since the economic demands and required environment conservation to remain in competitive markets.

This paper presents a new solution to diagnose business processes. In order to do that, the monitorization of a running workflow instance takes place. But this monitorization is only linked to some checkpoints, that must be determined previously. By means of this monitorization and the execution of a choreography analysis, an equivalent structure is obtained offline, which will be used to perform both local and global diagnosis, getting a high computational efficiency. This equivalent structure represents the relations among the different business processes and their external interactions, so that the local and global diagnosis can be performed without any information about the internal interactions of the different activities.

The diagnosis of the business processes will be carried out from information obtained from the checkpoints, indicating whether the events of the processes are correct or incorrect.

The business process lifecycle consists of four phases related to each other and organized in a cyclical way, where their dependencies can be detected. Fig. 1 represents the lifecycle of a business process. In the *Design and Analysis* phase, an analysis of the environment takes place, identifying needs and defining requirements, and finally designing and modelling the business process. In the *Configuration* phase, the business process is implemented, and the project preparation and realization is done. In the *Enactment* phase, an execution or deployment of the business process is carried out. Finally, in the *Evaluation* phase, the business process is monitored. The diagnosis process is included within the enactment phase, where after the execution of the different activities it is possible to detect an abnormal behavior of the choreography.

The kind of process choreography to diagnose S is formed by a finite set of business processes $\{BP_1, BP_2, ..., BP_n\}$, which are related among them using external interactions $\{s1, s2, ..., sp\}$. Also, the choreography has start events $\{i1, i2, ..., iq\}$ and end events $\{o1, o2, ..., or\}$. This implies that, in the business process under study there are three types of information: start and end events; internal interactions (only used within each business process) and external interactions (events among activities of different business processes). The external information is necessary to perform the local diagnosis.

Each business process (BP_i) is formed by a set of activities $(\{A_{i1}, A_{i2}, \ldots, A_{ik}\})$. The activities of a business process interchange incoming and outgoing interaction information. The incoming interaction information represent the inputs and the outgoing interaction information contains the results of the different activities. Both kind of interaction information are the internal and external interactions with activities in the same or in a different business process respectively.

In local and distributed diagnosis, the concepts of *Context Sets* and *Clusters* were defined [2], and in this paper these ideas are adapted for business process diagnosis:

Definition 2.1. *Context Set (CS)*: Any subset of activities of a business process. There are $2^{nactiv} - 1$ possible context sets, where *nactiv* is the number of activities of the business process.

Defining a business process as a directed graph, where nodes represent the different activities, and the directed edges are the internal and external interactions:

Definition 2.2. *Cluster (C)*: A connected component of the graph. The clusters of each business process will be taken into account in the local diagnosis process.

In our proposal, the process choreography (S) has an oracle which checks the end events, that are the checkpoints, finding out if they are correct (OK) or incorrect (KO).

Each business process involved in the process choreography has its own set of activities and requirements. Furthermore, each business process has associated an agent, which is in charge of the propagation and diagnosis tasks:

Definition 2.3. $SAgent_i$: Agent associated to BP_i that process incoming interaction information, performs local diagnosis and sends outgoing interaction information.

3 Motivating Example

In order to explain our proposal, we are going to use the example shown in Fig. 2, where there are four business processes, *(BP_1, BP_2, BP_3 and BP_4)*.

In the example, the different business processes with their respective information are:

- $\boldsymbol{BP}_1$ with the start event $\{i1\}$ and {s1, s2} as external interactions.
- $\boldsymbol{BP}_2$ with the start event $\{i2\}$ and {s1, s2, s3, s4, s5, s6, s7} as external interactions.
- $\boldsymbol{BP}_3$ with no start events and {s3, s4, s5, s6, s7, s8, s9, s10, s11} as external interactions.

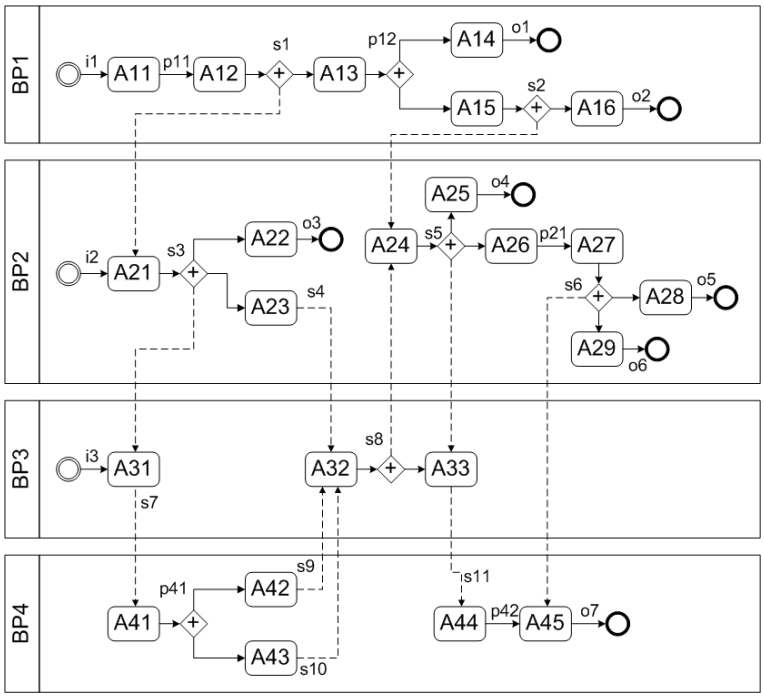

Fig. 2: Example of Business Processes.

– $\boldsymbol{BP_4}$ with the start event $\{i3\}$ and {s6, s7, s9, s10, s11} as external interactions.

The rest of the interactions in the processes and the activities are internal, and the are only known by the own process internally.

Each business process has a table where the relation between the non internal events and the rest of process (BP_1, BP_2, BP_3 and BP_4 for our example) is stored. That is, each business process knows how it is connected to other business processes by means of the different external interactions. Each business process does not know how the rest of the business processes are formed, only knows from what business processes have to receive incoming interaction information and towards what business processes have to send outgoing interaction information.

Once all the business processes know whether they are related to any incorrect output events, the diagnosis is solved locally in each business process (selfdiagnosis for each business process).

4 Logic and Distributed Algorithm

The *SAgent* of each business process connected to the end events of the receives information from an oracle about what end events are correct or incorrect. Based on this information, an algorithm which involves the process choreography takes place, and each *SAgent* can determine the local diagnosis that can be merged with the rest of the business processes. To carry out the steps of this algorithm, it is necessary to send interaction information among the different *SAgents*. For this reason, we suppose that the sending of information is always possible. In general, the distributed diagnosis process has four different phases to obtain the global diagnosis. Although we are going to use an example to explain all the relevant details, the main steps of the algorithm are:

– **Determining the local clusters:** In each business process involved in the process, an offline choreography analysis is executed by the *SAgent*. The results of this step are the different clusters (according to definition 2.2) of each business process, that form an equivalent structure which is used in the rest of the steps to improve the temporal efficiency of the diagnosis process.
– **Receiving the oracle:** Each *SAgent* related to the end events ($\{o1, o2, o3, o4, o5, o6, o7\}$ in the example shown in Fig. 2) receives incoming interaction information from the oracle with the information about the outputs that fail.
– **Propagation phase:** When a *SAgent* receives incoming interaction information with the information about what external interactions can be failing, an internal algorithm to decide which interactions could be correct or incorrect takes place. As a result of this algorithm, each *SAgent* sends outgoing interaction information to the *SAgents* of its neighbor business processes informing about the correct and incorrect interactions detected. This interaction information will travel from the business processes related to the end events ($\{o1, o2, o3, o4, o5, o6, o7\}$ in the example) to the business processes that receive the start events ($\{i1, i2, i3\}$ in the example).
– **Local diagnosis phase:** Depending on the received interaction information, each *SAgents* has to decide if the local diagnosis is necessary.

In order to improve the computational complexity, the first step of the algorithm to determine the local clusters is perform offline and only once, so that the next steps are based on the precompiled structure to carry out the diagnoses.

The steps of this algorithm are represented in Fig. 3, and will be explained in more detail using the example shown in Fig. 2.

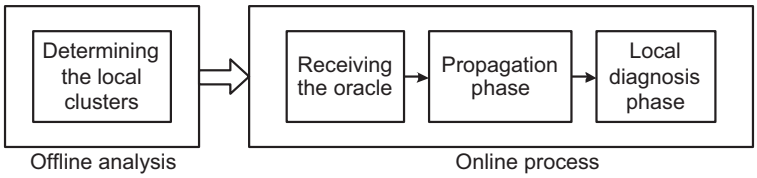

Fig. 3: Steps of the algorithm.

4.1 Determining the Local Clusters

In order to obtain the clusters of each business process, a choreography analysis takes place. Each *SAgent* derives the local clusters, which are formed by a set of activities linked through external and internal interactions (structural dependence), and not connected through internal interactions with the activities of a different cluster.

To know if an activity is working correctly is possible analyzing their outputs, inputs or other activities related to it by external or internal interactions. Within each cluster, there exist internal subsets (*IS*). The idea of an IS is to build a set of activities where the external and internal interactions are in more than one activity to derive the diagnosis using choreography analysis. In order to clarify it, new notations are used:

External(A_{ij}) are the external interactions (shared information) for the activity A_{ij}
Internal(A_{ij}) are the internal interactions (private information) for the activity A_{ij}

Definition 4.1. *Internal Subset (IS)* for a cluster in a business process BP_i is a set of activities, where for each one of its activities A_{ij}:

$$\begin{aligned}
&\forall\ \mathrm{v} \mid \mathrm{v} \in \mathit{Input}(A_{ij}){:}\ \mathrm{v} \in \{\mathrm{i}1, \ldots, \mathrm{i}q\} \\
&\quad \vee\ (\mathrm{v} \in \mathrm{Internal}(A_{ij}) \wedge \exists\ A_{ik} \neq A_{ij} \mid A_{ik} \in \mathrm{IS} \mid \mathrm{v} \in \mathrm{Output}(A_{ik})) \\
&\quad \vee\ (\mathrm{v} \in \mathrm{External}(A_{ij}) \wedge \exists\ A_{ik} \neq A_{ij} \mid A_{ik} \in \mathrm{IS} \mid \mathrm{v} \in \mathrm{Output}(A_{ik})) \\
&\quad \vee\ (\mathrm{v} \in \mathrm{External}(A_{ij}) \wedge \mathrm{v} \in \mathrm{External}(A_{jk}) \mid A_{jk} \in \mathrm{BP}_j \neq \mathrm{BP}_i) \\
&\forall\ \mathrm{v} \mid \mathrm{v} \in \mathit{Output}(A_{ij}){:}\ \mathrm{v} \in \{\mathrm{o}1, \ldots, \mathrm{o}r\} \\
&\quad \vee\ (\mathrm{v} \in \mathrm{Internal}(A_{ij}) \wedge \exists\ A_{ik} \neq A_{ij} \mid A_{ik} \in \mathrm{IS} \mid \mathrm{v} \in \mathrm{Input}(A_{ik})) \vee (\mathrm{v} \in \mathrm{External}(A_{ij}))
\end{aligned}$$

and for the set of activities that form each internal subset IS:

$$\begin{aligned}
&\forall\ \mathrm{v} \mid \mathrm{v} \in \mathit{Input}(IS){:}\ \mathrm{v} \in \mathrm{Input}(A_{ij}) \wedge A_{ij} \in IS \wedge (\ \mathrm{v} \in \{\mathrm{i}1, \ldots, \mathrm{i}q\} \vee \mathrm{v} \in \mathrm{External}(A_{ij})) \\
&\forall\ \mathrm{v} \mid \mathrm{v} \in \mathit{Output}(IS){:}\ \mathrm{v} \in \mathrm{Output}(A_{ij}) \wedge A_{ij} \in IS \\
&\quad \wedge\ (\ \mathrm{v} \in \{\mathrm{o}1, \ldots, \mathrm{o}r\} \vee (\ \mathrm{v} \in \mathrm{External}(A_{ij}) \wedge \nexists\ A_{ik} \neq A_{ij} \mid A_{ik} \in IS \wedge \mathrm{v} \in \mathrm{External}(A_{ik})))
\end{aligned}$$

For the example, the different clusters calculated by the *SAgents* are:

- $SAgent_1$: {$A11$, $A12$, $A13$, $A14$, $A15$ and $A16$}
- $SAgent_2$: {$A21$, $A22$ and $A23$} and {$A24$, $A25$, $A26$, $A27$, $A28$ and $A29$}
- $SAgent_3$: {$A31$} and {$A32$ and $A33$}
- $SAgent_4$: {$A41$, $A42$ and $A43$} and {$A44$ and $A45$}

For example, the activities {$A21$, $A22$, $A25$} form a new cluster with three internal subsets as it is shown in Fig. 4.

This new cluster works as a black box for the rest of the *SAgents*, and its activities can be connected with other clusters by external interactions.

In general, the different *SAgents* form a new system where the external interactions establish the local connection between the different business processes, as it is shown in Fig. 5. This obtained structure will be used in the rest of the algorithm to carry out the diagnosis process. This structure, customized for the diagnosis process, is prepared offline, and will be used to perform the local diagnosis phase, building a table with the relations between the activities and the external interactions that compose each business process.

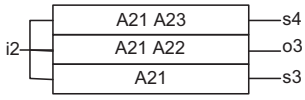

Fig. 4: A local cluster with three ISs.

4.2 Receiving the Oracle

The process choreography depends on an oracle which is able to determine the end events that can be failing.

Each *SAgent* related to the end events ($\{o1, o2, o3, o4, o5, o6, o7\}$ in the example shown in Fig. 2) receives interaction information from the oracle with the information about all the end events, indicating whether they are wrong or correct. In case of all the end events are correct, it is not necessary any propagation or diagnosis, because all the activities are working correctly. When at least an end event is wrong, the process to determine what activity does not work correctly starts. The diagnosis process starts in this step.

4.3 Propagation Phase

In order to explain next steps of the algorithm, these new definitions have to be introduced:

Definition 4.2. *Possible Incorrect Event (PIE):* It is an external interaction related to an incorrect end event. An interaction is related to an end event if when the interaction changes, the end event also changes.

Definition 4.3. *Correct Event (CE):* It is an external interaction related to a correct end event. If an interaction is related to a correct end event and an incorrect end event simultaneously, the interaction is defined as a *Correct Event*, since we suppose that two incorrect activities cannot generate a correct end event.

Depending on the end events, it is possible to know if the process choreography is working correctly. In order to know which activities are failing, the external interactions between the business processes are used to infer which interactions are failing. For example, for the local cluster shown in Fig. 4, being $s3$ a *PIE* and $s4$ a *CE*. It means that the activity $A21$ is failing, but it is not possible because if the external interaction $s4$ is correct, the activity $A21$ would be correct. It means that the external interaction $s3$ considered as a *PIE* actually

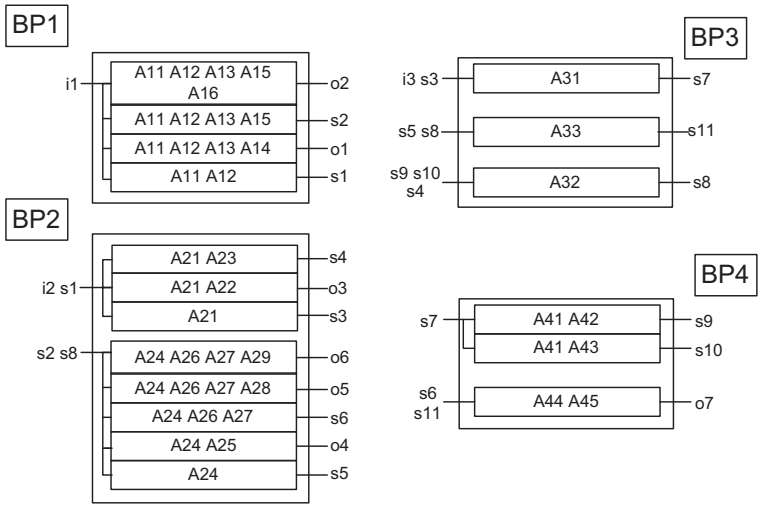

Fig. 5: Clusters and ISs of the process choreography.

it is not an incorrect event. If an interaction is correct all the activities associated to the IS of this interaction are correct. A PIE will be an IE if it has at least one activity that is not a correct activity. This idea is described in the following definition:

Definition 4.4. *Incorrect Event (IE):* Given the set *ExternalInteractions(IS_i)* that are output events of IS_i, `e` is an IE if it is a *PIE* and $\exists\ A_{kl} \in \texttt{IS}_i \mid \forall\ IS_j : j \in 1 \ldots n \wedge i \neq j \nexists\ \texttt{e'} : \texttt{e' is a}\ CE \wedge \texttt{v'} \in \texttt{ExternalInteractions(}IS_j\texttt{)} \wedge A_{kl} \in IS_j$

On this step of the algorithm, some interaction information is exchanged between the business processes. This is information about the *CE*s and *PIE*s, and the traffic direction goes from the clusters related to the end events {o1, ..., or} to the clusters which provide them the start events {i1, ..., iq}.

Therefore, according to the previous definition, the definition of *CE* is reinforced with this new concept, since a *CE* is also a event that is not an Incorrect Event.

In the previous step of the algorithm, the oracle sends the evaluation of the end events to the *SAgents* with clusters related to them, indicating which ones are correct (OK) or incorrect (KO). In this step these *SAgents* are able to infer what external interactions are *CE*s or *IE*s according to definition 4.4, and to send this interaction information to the business processes to which they are linked through their input events.

In order to do that, each *SAgent* needs to receive the information about all the outputs os a cluster before inferring whether the external interactions of that cluster are *CE*s or *IE*s. Therefore, the propagation will take place as the information of the outputs of the clusters is known by the *SAgents*. This is not a linear process since several business process can have clusters with a dependency between them. This is, for example, if a business process BPa has the clusters Cx and Cy, and another business process BPb has the cluster Cz, it is possible that Cx depends on Cz, that depends on Cy. In this example, to carry out the propagation, the *SAgent* of BPa waits for all the information about the outputs of Cx to propagate to Cz. In the same way, the *SAgent* of BPb will propagate from Cz to Cy when it has all the information about Cz. So, both *SAgents* have interchanged interaction information in both directions.

The interaction information sent by the *SAgents* are of the form shown in Fig. 6.

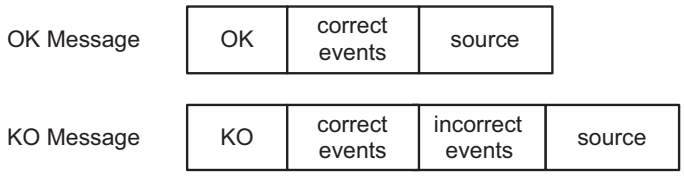

Fig. 6: Format of the interaction information.

These messages carry the following information:

- **OK/KO** field: type of information.
- **Correct events** field: the external interactions labelled as *CE*s by the *SAgent* of the business process source of the interaction information.
- **Incorrect events** field: the external interactions labelled as *PIE*s by the *SAgent* of the business process source of the interaction information.
- **Source** field: indicates the source of the interaction information. It can be a neighbor business process or the oracle to indicate the beginning of the diagnosis process.

The procedures in the algorithms 1, 2, 3 and 4 describe the behavior of a *SAgent* after receiving an OK or a KO interaction information.

Algorithm 1: Receiving a KO interaction information

```
Input: interactionInformation(KO, X_OK, X_KO, BP_i)
begin
    label as CEs the external interactions which are in X_OK;
    foreach external interaction e in X_OK do
        if e is not labelled as CE then
            label e as PIE;
    if the SAgent has received the incoming interaction information from all the
    business processes related to the outputs of the cluster C then
        checkEvents(C);
        propagateInformation(C);
    if the SAgent has not received the incoming interaction information from all the
    business processes related to its outputs then
        wait for the rest of the incoming interaction information;
end
```

Algorithm 2: Receiving an OK interaction information

```
Input: interactionInformation(OK, X_OK, BP_i)
begin
    label the external interactions which are in X_OK as CEs;
    if the SAgent has received the incoming interaction information from all the
    business processes related to the outputs of the cluster C_x then
        checkEvents(C_x);
        propagateInformation();
    if the SAgent has not received the incoming interaction information from all the
    business processes related to its outputs then
        wait for the rest of the incoming interaction information;
end
```

Algorithm 3: Procedure for labelling events as correct or incorrect

```
checkEvents(Cluster C):
begin
    /*definition 4.4*/
    foreach event e labelled as a PIE do
        if C ∈ IS_i and A_kl is an activity of IS_i and not exists another IS_j that
        contains A_kl with an event e' labelled as a CE and e' is different from e and e'
        belongs to Output(IS_j) then
            label e as an IE;
        else
            label e as a CE;
    foreach output event e labelled as IE do
        label as PIE the events which are inputs of the ISs where e is an output;
end
```

Algorithm 4: Procedure for propagating information

```
propagateInformation (Cluster C):
begin
    foreach BP_i related to C by the set of interactions X do
        if exists a subset of events X_i which are inputs of C and outputs of BP_i and
        are labelled as PIEs then
            send the interaction information (KO,X − X_i, X_i, this) to the SAgent_i;
        else
            send the interaction information (OK, X, this) to the SAgent_i;
end
```

The different clusters are connected by means of external interactions. For example, if $o5$, $o6$, and $o7$ are KO, and the rest of the end events are OK, the sent information is as follows:

- The oracle sends interaction information to $SAgent_1$, $SAgent_2$ and $SAgent_4$ indicating that $o5$, $o6$, and $o7$ are the incorrect end events.
- $SAgent_4$ has all the information about the outputs of one of its clusters. Therefore, it labels $o7$ as IE and propagates to $SAgent_3$ and $SAgent_2$ that s11 and s6 are PIEs.
- $SAgent_3$ has all the information about the outputs of one of its clusters ($s11$ is PIE). It propagates to $SAgent_2$ that $s5$ and $s8$ are incorrect.
- $SAgent_2$ has all the information about its biggest cluster ($o5$, $o6$, $s5$ and $s6$ are incorrect, and $o4$ is OK). According to definition 4.4, $s2$ and $s8$ are CEs, and this is the information propagated to $SAgent_3$ and $SAgent_1$.
- $SAgent_3$ had different kinds of information about $s8$ (CE and PIE simultaneously), therefore $s8$ is labelled as a CE. $SAgent_3$ can determine that $s4$, $s9$ and $s10$ are CEs.
- etc.

This process finishes when all the $SAgents$ have propagated all the interaction information about the inputs of all their clusters, so that the local diagnosis phase can start.

4.4 Local Diagnosis Phase

Using the information collected from the previous step, the local diagnosis is executed by the $SAgents$ which have any IE events.

The local diagnosis process has two phases (offline and online):

(i). **To build a signature matrix:** With the information obtained from Section 4.1, a signature matrix is created. This matrix relates each external interaction and end events of the business process with the activities of the ISs where this interaction participates. The matrix has *number of interactions and end events* rows and *number of activities* columns. This process is done only once, storing precompiled information. The construction of the matrix is an adaptation of the algorithm presented in [3] for business processes. An example is shown in Fig. 7(a) that represents the signature matrix for one of the clusters of $BP2$ in the example.

(ii). When the IEs are known, only the part of the matrix related to the possible wrong activities is analyzed. It means that only the rows of the IEs and the activities non related to any CE will participate in the local diagnosis. With this subset of relations among interactions, end events and activities, the set of activities which are not working correctly must be found. According to diagnosis theory, the best way to do that is calculating the hitting sets and minimal hitting sets, whose definition is as follows:

Definition 4.5 *Hitting Set (HS)* for a collection of components $\mathcal{C}$ is a set of components $\mathcal{H} \subseteq \bigcup_{\mathcal{S} \in \mathcal{C}} \mathcal{S}$ such that $\mathcal{H}$ contains at least one element for each $\mathcal{S} \in \mathcal{C}$. A HS of $\mathcal{C}$ is

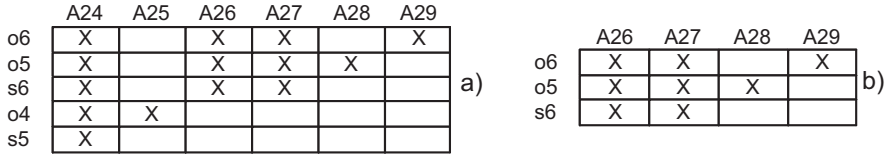

Fig. 7: Signature matrix of a cluster of $BP2$.

minimal iff no proper subset of it is a HS of $\mathcal{C}$. The minimal HSs for a set of sets are formed by $\{\mathcal{H}_1, \mathcal{H}_2, \ldots \mathcal{H}_n\}$, where $\mathcal{H}_i$ is a minimal HS of components. The cardinality of $\mathcal{H}_i$ ($|\mathcal{H}_i|$) is the number of components of $\mathcal{H}_i$. This definition can be adapted to activities instead of components.

In the case of the matrix, each set will be formed by the activities of each row of the matrix related to any IE, so that $\mathcal{C} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots \mathcal{S}_n\}$, where each $\mathcal{S}_i$ is the set of software processes related to each IE. In Fig. 7(b) appears the part of the signature matrix of the cluster analyzed to perform the local diagnosis if $o5$ and $o6$ are KO and, therefore, $s6$ is IE.

Finally, the minimal hitting sets are the diagnosis of the business processes. For the example in Fig. 7(b) there are two minimal hitting sets, which are $\{A26\}$ and $\{A27\}$.

(iii). The local diagnosis can be improved storing all the final diagnosis according to the IEs. It means that if an IE has already been analyzed, the diagnosis will be very efficient. Each *SAgent* counts on a local list where it stores the fault signature from the local diagnosis executed until that moment, so that when a *SAgent* has to diagnose its activities, previously it checks whether it has already done the same diagnosis before.

Each element of the lists contains two fields with the next information:

(a) External interactions labelled as IEs.

(b) Activities which can fail according to the IEs detected.

The local diagnosis results depend on the subset of external subsets which are IEs, since two different analysis with the same set of IEs will obtain the same minimal hitting sets of activities. Using a local list to store previous results makes possible a faster process of diagnosis, because of the reutilization of the information obtained so that it is not necessary to carry out the same analysis twice.

Finally, the global diagnosis D_g can be seen as the union of all the local diagnosis: $D_g = D_1 \cup D_2 \cup ... \cup D_n$, being n the number of business processes.

According to our example, Table 1 shows some example results of the self-diagnosis on each business process, presenting what activities can be failing.

Table 1: Diagnosis Results for each Business Process

KO end events	BP1	BP2	BP3	BP4
{o5, o6, o7}	-	{A23} {A26} {A27}	{A31} {A32}	{A41} {SA44} {A45}
{o2, o4}	{A16}	{A25}	-	-
{o3, o5, o7}	-	{A21} {A28}	{A31} {A32} {A33}	{A41} {SA44} {A45}
{o1, o3, o4}	{A14}	{A22} {A25}	-	-
{o1, o2, o6}	{A14, A15} {A14, A16}	{A29}	-	-

5 Conclusions and Future Work

In this work how to perform the diagnosis of business processes is presented. These business processes work with public and private information (external and internal interactions), and the diagnosis process can be performed without needing to know neither all the information nor the business process model. By means of the use of previously compiled knowledge, a high temporal efficiency is obtained, since the preparation of an equivalent structure is performed offline.

Our proposal of fault diagnosis is an innovative solution to the diagnosis of business processes, although it does not find the global minimal diagnosis, since the set of activities found as a solution is not a global minimal hitting set.

As future work, we will work in the detection of other kinds of failures in business processes, such as the errors derived from the creation of a model that does not correspond to the real problem, or errors in the definition of the interaction between the activities of a process, or between the activities of different processes.

Acknowledgment

This work has been partially funded by the Ministry of Education and Science of Spain (DPI2006-15476-C02-01) and European Regional Development Fund(ERDF/FEDER).

References

1. R Bianchini and R Buskens, Implementation of on-line distributed system-level diagnosis theory.
2. D Borrego, M T Gómez-López and R M Gasca, Diagnosing distributed systems using only structural and qualitative information. International Transactions on Systems Science and Applications (to appear), 2008.
3. R Ceballos, M T Gomez-Lopez, R M Gasca and C del Valle, A compiled model for faults diagnosis based on different techniques. AI Commun., Vol. 20, No. 1, 2007, pp. 7–16.
4. R Debouk, S Lafortune and D Teneketzis, Coordinated decentralized protocols for failure diagnosis of discrete-event systems.
5. E Fabre, A Benveniste and C Jard, Distributed diagnosis for large discrete event dynamic systems. 15th IFAC World Congress, Barcelona 2002.
6. P Frohlich, I de Almeida Mora, W Nejdl and M Schroeder, Diagnostic agents for distributed systems. ModelAge Workshop 1997, pp. 173–186.
7. M T Gomez-Lopez, R M Gasca, C D Valle and S Pozo, Distributed model-based diagnosis using object-relational constraint databases.. AINA (2) 2006, pp. 866–870.
8. X L Guillou, M O Cordier, S Robin and L Rozé, Chronicles for on-line diagnosis of distributed systems. Research Report. ftp://ftp.irisa.fr/techreports/2008/PI-1890.pdf 2008.
9. J D Kleer, A Mackworth and R Reiter, Characterizing diagnoses and systems. Artificial Intelligence 56, Vol. 2-3, 1992, pp. 197–222.
10. Y Pencole, Decentralized diagnosier approach: application to telecommunication networks. 13th International Workshop on Principles of Diagnosis 2000, pp. 185–192.
11. G Provan, A model-based diagnosis framework for distributed systems.
12. R Reiter, A theory of diagnosis from first principles. Artificial Intelligence 32, Vol. 1, 1987, pp. 57–96.
13. I Roychoudhury, G Biswas, X Koutsoukos and S Abdelwahed, Designing distributed diagnosers for complex physical systems.. 16th International Workshop on Principles of Diagnosis 2005, pp. 31–36.
14. R Su and W M Wonham, A model of component consistency in distributed diagnosis.. 15th International Workshop on Principles of Diagnosis 2004, pp. 62–68.
15. M Weske, Business Process Management: Concepts, Languages, Architectures, Springer-Verlag, 2007.