DECENTRALIZED DIAGNOSIS FOR BPEL WEB SERVICES

Lina Ye and Philippe Dague
LRI, Univ. Paris-Sud, CNRS and Gemo, INRIA Saclay, Parc Club Orsay Université
4 rue Jacques Monod, bât G, Orsay, F-91893, France

Keywords: BPEL Web services, Petri nets, consistency-based diagnosis, causal model, decentralized diagnosis.

Abstract: This paper proposes a decentralized diagnosis approach for BPEL Web services, where a local diagnoser is provided to each BPEL service and should cooperate with a coordinator. It should noted that there is no global model in this approach but local model for each service. The strategy of local consistency-based diagnosis consists in abstracting diagnostic knowledge base from data dependencies contained in the enriched BPEL Petri net model of each service and then reasoning from the observations with a set of local possible source faults. The coordinator has to put together all the local diagnosis information from local diagnosers and infer global diagnoses.

1 INTRODUCTION

Self-healing software is one of the important challenges for Information Society Technologies research. Our paper proposes a decentralized diagnosis approach for BPEL Web services, in the context of the EU project WS-Diamond (http://wsdiamond.di.unito.it), whose goal is to design a framework for self-healing Web services by adopting artificial intelligence methodologies to solve the diagnosis problem by supporting online detection and identification of faults.

Considering a complex Web service, due to its data oriented nature, we focus on data semantic faults, which are the most difficult to identify but the most critical by their consequences since Web services are components that use messages to interact with each other and messages are mostly a data structure. In fact, many subtle faults on data cannot be identified at the time when they happen and are thus difficult to diagnose. Generally, data are propagated through the interactions and then are used to elaborate other data or control decisions, like branching conditions. In other words, during the transmission through partner instances, data may propagate a dysfunction until an exception is thrown when a logical data contradiction is detected. The contradiction can be either a mismatch of some data values or some wrong data. This kind of dependency makes it difficult to predict the relationship between the source fault and the symptom. For example, let us consider a data fault caused by a discrepancy in the semantic of the date format: in a travel agency service, a customer defines his itinerary by French format, like from $(6/2/07)$ to $(10/2/07)$ for hotel reservation, while hotel service uses English format for interpretation. Then the user will raise an exception, saying that the amount of the hotel cost is exaggerated (4 months instead of 4 days). To detect and explain such kind of faults, model-based diagnosis (Hamscher et al., 1992) is adopted here owing to its capability of detecting more effectively the unanticipated or hidden faults in the context of Web services. However, it should be noted that Web services are a novel area of applications for model-based diagnosis and no such characterization is available from the literature.

The rest of this paper is organized as follows. Section 2 briefly presents the modeling of BPEL(Business Process Execution Language) services. In section 3, a decentralized diagnosis approach is proposed, including the strategy of local diagnosis based on consistency-based perspective and a protocol for global diagnosis. At last the conclusion is given in section 4.

2 BPEL SERVICES MODELING

BPEL is a standard Web service composition language, defined to support the development of com-

Ye L. and Dague P. (2008).
DECENTRALIZED DIAGNOSIS FOR BPEL WEB SERVICES.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 283-287
DOI: 10.5220/0001525402830287

plex applications based on the orchestration of simpler ones. It offers a set of structured activities to order the execution of the basic ones and thus control the process flow. (Li et al., 2007) has already precisely described a method for generating automatically enriched BPEL Petri net models with data dependencies from their BPEL code. The essential idea is to use places to represent the most elementary parts that compose a message (using its Xpath decomposition) as well as the activation status and to use transitions to represent activities. More importantly, to enrich each transition of the BPEL Petri net with a set of dependency relations between its input and output parameters, three data dependency relations for Web service diagnosis at high level are considered, defined in (Ardissono et al., 2005): $FW(a,x,y)$ describes the relation that the output variable y of the activity a is a copy of the input variable x, regardless of the correctness of the behavior of a; $SRC(a,y)$ means that the output variable y is created by the activity a, independently of its input variables; $EL(a,\{x_1,x_2...x_k\},y)$ expresses the case that the output variable y is computed from the input variables $x_1,x_2...,x_k$ by the activity a. By modeling BPEL services in this way, it is possible to get all data dependencies in services such that our diagnosis result concerns all possible source faults.

3 DECENTRALIZED DIAGNOSIS FOR BPEL SERVICES

Web services, as all other applications, are subject to dysfunctions, such as a faulty composed service, an incoming message mismatching the interface, etc. The symptom is that exceptions are thrown at the places where the process cannot be executed. To handle these faults, the current mechanism is a throw-and-catch one, which is very preliminary since it relies on the empirical knowledge of the developer while various causes of the exception may be unknown to him. So, with this mechanism, we cannot obtain a sound and complete diagnosis, especially for semantic faults. For example, when an exception is thrown, then not only the service that throws the exception should be suspected, but the one that generates the data and also all the services that modify the data should be suspected. Whereas a current Web service exception can only report about where the exception happens. As said before, since the semantic faults are the most difficult and the most critical ones to diagnose, our approach focus on determining the exact source faults that are possibly responsible for this kind of exceptions. For this, three types of source faults, which may cause semantic faults in Web services, are taken into account: a basic Web service (black box) with abnormal behavior; a wrong input data from the user; and an interface fault, which means that the shared messages between two cooperating Web services become different and thus the dysfunction can be engendered as a result.

3.1 Architecture

We propose a decentralized diagnosis approach in this paper, considering its efficiency in Web services context. Figure 1 describes our architecture. For each BPEL service W_i, there is a local diagnoser D_i performing local diagnosis inference by adopting consistency-based approach, based on the model of W_i as well as local observations, that are logged by the monitoring platform. In addition, there is also a coordinator, which is considered as a separated Web service that receives local diagnosis results from local diagnosers to decide about the global diagnoses by resolving local diagnosis conflicts. Here it is worth noting that there is no global model for the whole BPEL process, since the coordinator knows nothing about local models but their connections. In this sense our approach is called a decentralized one. Obviously, in

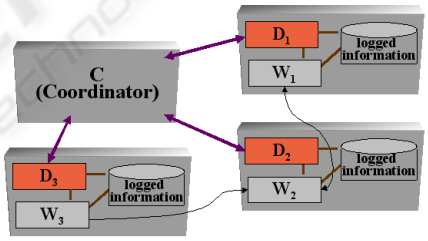

Figure 1: Decentralized diagnosis architecture.

this architecture, for the organization owning the orchestrated Web service, the privacy issues can be satisfied due to the fact that the model of W_i can only be inspected by the associated local diagnoser D_i, which does not directly interact with other local diagnosers.

3.2 Local Diagnosis for BELL Services

Reiter's logical theory of diagnosis (Reiter, 1987) is also referred to as consistency-based diagnosis. In this approach, a behavioral system model is defined as a tuple $(SD,COMP)$: SD, the system description, is constituted of a finite set of first order sentences describing explicitly the behavior of each given component; $COMP$ is a finite set of constants, meaning the components that could be faulty. In SD, a distinguished predicate $'ab'$ means abnormal. For example, $ab(c)$ denotes an abnormal component c, while

$\neg ab(c)$ expresses the component c working correctly. An observed behavioral system model is expressed as a tuple $(SD, COMP, OBS)$, where $(SD, COMP)$ is a system model and OBS is a set of first order formulas expressing the observations. Generally, a diagnosis is considered as a hypothesis that certain components of the system are behaving abnormally, which should be consistent with what is known about the system and with the observations. A diagnosis is called minimal diagnosis, if and only if no proper subset of it is a diagnosis at the same time. Usually minimal diagnoses are often the preferred ones, considering the principle of parsimony.

Definition 1. *A minimal diagnosis for an observed system (SD, COMP, OBS) is a minimal subset $\triangle$ of COMP such that:*

$$SD \cup OBS \cup \{ab(c) | c \in \triangle\} \cup \{\neg ab(c) | c \in COMP \setminus \triangle\}$$

is consistent.

The above behavioral system model description SD can be simplified by using causal system model description SD' (Bauer, 2005). Differently from SD, SD' describes the causal relation that if all input data and the component/activity itself are correct, then all its outputs are correct. However, this kind of causal relations are not precise enough. For Web service diagnosis, owing to the availability of the three dependency relations, described in 2, we can get a similar but more precise system description, denoted as SD''. Specially, the causal relations in SD'' are obtained from every data dependency(FW, SRC, EL), while the causal relations in SD' are acquired only from every component/activity.

Since we have a set of data dependencies in the enriched BPEL Petri net model, it is crucial to transform them into causal logic formulas for the consistency-based local diagnosis. Tables 1, 2, 3 describe the causal relations between input and output parameters for all kinds of activities by translating each data dependency into causal logic formula. Considering table 1, it can be seen that $\neg ab(a)$ in the logic formula is not concerned. The reason is that in our case, BPEL code is supposed to be correct, which means that the behavior of basic BPEL activity is assumed to be normal, since we focus only on semantic faults and not on program debugging. In addition, due to the fact that when an exception is thrown, multiple causes could be possible to explain this exception, depending on which branch being taken, we adopt online diagnosis to infer the exact possible sources of the fault. For this, $obs(a)$, which expresses that basic BPEL activity a is observed, should be added into the logic formulas. In this way, the logged information of our monitoring system, like the observed activities, can be used in the diagnosis process. For example, for the dependency $SRC(a,y)$, the corresponding logic formula means that if input activation status of activity a is correct and a is observed, then output y is correct. Similarly, table 2 can be obtained. The difference is that for Invoke activity, we have to take the invoked service into account for SRC and EL dependencies, since the abnormal behavior of the invoked service affects the output parameters. Here for the sake of simplicity, the correct behavior of invoked service, invoked by Invoke activity a, is denoted as $\neg ab(a)$. Table 3 is for an additional type of activity, control activity, which is not a basic BPEL activity and thus cannot be observed by our monitoring system. So in 3, $obs(a)$ is not concerned. It can be noticed that control activity has no output data place but only one output activation place, meaning that the role of control activity is just to control the process flow.

Table 1: The transformation for basic BPEL activity except Invoke.

Dependencies	Causal logic formulas
$FW(a,x,y)$	$\neg ab(a.in) \wedge \neg ab(x) \wedge obs(a) \rightarrow \neg ab(y)$
$SRC(a,y)$	$\neg ab(a.in) \wedge obs(a) \rightarrow \neg ab(y)$
$EL(a,\{x_1,x_2\},y)$	$\neg ab(a.in) \wedge \neg ab(x_1) \wedge \neg ab(x_2) \wedge obs(a) \rightarrow \neg ab(y)$

Table 2: The transformation for Invoke activity.

Dependencies	Causal logic formulas
$FW(a,x,y)$	$\neg ab(a.in) \wedge \neg ab(x) \wedge obs(a) \rightarrow \neg ab(y)$
$SRC(a,y)$	$\neg ab(a.in) \wedge \neg ab(a) \wedge obs(a) \rightarrow \neg ab(y)$
$EL(a,\{x_1,x_2\},y)$	$\neg ab(a.in) \wedge \neg ab(x_1) \wedge \neg ab(x_2) \wedge \neg ab(a) \wedge obs(a) \rightarrow \neg ab(y)$

Table 3: The transformation for control activity.

Dependencies	Causal logic formulas
$FW(a,a.in,a.out)$	$\neg ab(a.in) \rightarrow \neg ab(a.out)$
$EL(a,\{a.in,x\}, a.out)$	$\neg ab(a.in) \wedge \neg ab(x) \rightarrow \neg ab(a.out)$

For each enriched BPEL Petri net model of a BPEL service, a set of logic formulas, actually a set of Horn clauses, can thus be obtained from a set of data dependencies available in the model. This set of logic formulas is from now on called Diagnostic Knowledge Base (DKB). When applying the consistency-based diagnosis technique for complex physical systems to Web services, this DKB facilitates the calculation of the minimal diagnoses with OBS and with

the complementary knowledge of a set of possible sources whose faults can be used to explain the exception, denoted as PSF. Due to the assumption of correct BPEL code, we have the following definition:

Definition 2. *PSF is the set of possible source faults for a local BPEL service, concerning three types:*

- *Incorrect input data x from user, denoted as* $ab(x)$
- *Faulty basic Web service, invoked by Invoke activity a, denoted as* $ab(a)$
- *Wrong input variable y in Web service* W_i *coming from shared variable* y' *in another composite Web service* W_j*, denoted by* $ab(y)$.

For the last one, faulty input variable y from another service, it can be temporally considered as local possible source fault but will be exploited later. Details will be presented in 3.3. Then with DKB and PSF, similar to definition 1, we have:

Definition 3. *A minimal diagnosis for an observed BPEL service* (DKB, PSF, OBS) *is a minimal subset* $\triangle$ *of PSF such that:*

$$DKB \cup OBS \cup \{ab(c) | c \in \triangle\} \cup \{\neg ab(c) | c \in PSF \setminus \triangle\}$$

is consistent.

3.3 Protocol for Global Diagnosis

In our decentralized diagnosis architecture, each local diagnoser can interact both with its associated Web service and with the coordinator, while the coordinator can interact only with local diagnosers. Moreover, due to the privacy issues, our coordinator C does not initially have any information on the individual Web services except their connections, which are obtained offline and are at interface level.

This decentralized diagnosis process is started by a local diagnoser D_i that is awakened by an exception in Web service W_i . The whole steps are as the following:

1. D_i infers the local diagnoses for the observed system $\{DKB_i, PSF_i, OBS_i\}$ and sends its result to C.
2. C extends the diagnoses received from D_i by providing each element a of every diagnosis (here for the sake of generalization, minimal diagnosis could be single or multiple, which is thus considered as a list) with local diagnoser D_a that can further explain $ab(a)$. If there is no such local diagnoser, then D_a is assigned $null$. When the diagnosis is $\triangle = \{y\}$, where y is input data coming from y' in another composite Web service W_j, then the result of extension should be $\triangle' = \{Inf(y', y), null\}$ or $\triangle' = \{y', D_j\}$, due to the fact that abnormal y could be caused by the interface fault or by the local possible source fault in W_j. Here $ab(Inf(y', y))$ denotes the fault of interface between y in W_i and y' in W_j when the value is transmitted from y' to y. In particular, for $\triangle' = \{Inf(y', y), null\}$, the observed information can be used to verify or disapprove it. For example, if the values of the shared variables are observed different, then this interface fault is verified. Otherwise, it should be excluded. When the diagnosis is $\triangle = \{a\}$, a being input data from user or Invoke activity invoking a basic Web service, then it is extended as $\triangle' = \{a, null\}$, since no local diagnoser can be further required to explain $ab(a)$.
3. If there exist other local diagnosers for further request, C triggers one, for example D_j, by sending $ab(y')$ as an exception in W_j. D_j then performs local diagnosis and sends its result to C.
4. C extends the diagnoses from D_j and then makes diagnosis update by replacing $\{y', D_j\}$ with the extended result of D_j. After this, C continues to look for next local diagnoser. If there is any, the process turns to step 3. Otherwise, the process terminates. Finally, we can get all the global diagnoses from the final diagnoses in C.

Alg1 formally illustrates our diagnosis process.

Table 4: Meaning of major symbols in the algorithm.

Symbol	Meaning
H or H_a	a list of candidate minimal diagnoses, subsets of $(\cup_i PSF_i)$
ITF_i	a set of interfaces between shared variables in different services
ISV_i	a set of input variables in PSF_i from another composite Web service
list.next()	returns the first element of a list
list.remove(element)	removes an element from the list
list.add(element)	adds an element to a list
$D_i(Var)$	returns a set of local minimal diagnoses for W_i inferred by D_i with the exception on Var
list.extend()	extends a list, details precisely described in step2
H.update(a, H_a)	returns the result of replacing a with H_a in H

Alg 1. *Input: Variable* y_0 *in* W_i *where rises the exception. Output: F, the list of global minimal diagnoses* $\triangle_k$*, where* $\triangle_k$ *is a subset of* $((\cup_i PSF_i) \bigcup (\cup_i ITF_i) \setminus \cup_i ISV_i)$. *Initially, F is empty.*

$H = D_i(y_0)$;
$H.extend()$;
while $H != null$ ***do***
$\{T = H.next()$;

```
    if (for each element a in T, D_a = null), then
        {F.add(T);
         H.remove(T);}
    else
      for each element a in T, do
      if (D_a! = null), then
        {H_a = D_a(a);
         H_a.extend();
         H.update(a,H_a);}
  }
return F
```

4 CONCLUSIONS

In this paper, a cooperative decentralized diagnosis approach for complex Web services is proposed. BPEL Web services are chosen as our application due to their popularity and perspective. For each individual activity, grey box is adopted, which means that we do not model its internal behavior but the correlation between its input and output parameters. Thus we can infer how the correct/incorrect status of input parameters can affect the correct/incorrect status of output parameters. Obviously, our approach greatly reduces the computation complexity thanks to local diagnosis algorithm relying on Horn clauses inference and global diagnosis based on decentralized architecture. The details of our experimentations on real BPEL examples (in the framework of project WS-Diamond) are omitted due to lack of space. In addition, our approach can be easily extended to handle multiple exceptions, especially for independent exceptions in different paralleled branches. In this case, each exception should be diagnosed independently and then the synthesized diagnoses are the union of all the diagnoses. Since we focus on the minimal diagnoses, we remove the synthesized diagnoses that are supersets of other ones. Furthermore, it is straightforward to extend our diagnosis architecture to multi-layered hierarchies. For example, a coordinator can be designed to be able to act as a local diagnoser for another coordinator at higher level.

There are similar approaches in the literature. In (Bauer, 2005), the problem of contradicting first order system descriptions with observations is reduced to propositional logic, which is similar to our DKB. However, they have experienced k-satisfiability by using state-of-the-art SAT solvers to determine conflict sets and minimal diagnoses, which is avoided in our approach since our DKB is made up of Horn clauses and thus permits direct deduction. (Ardissono et al., 2005) has proposed a similar decentralized model-based diagnosis approach for Web services. Their global diagnoser does not initially have any information on the individual Web services such that the communications between local diagnoser and global diagnoser should contain the information about diagnosis and interactions (like connection information). Differently, our coordinator has the knowledge about the connections between services, which lightens the communication flow since just diagnosis information should be considered. However, this knowledge does not violate the privacy issues since it is at interface level and thus the coordinator still does not know the internal details of services. In addition, they just have proposed the characterization of local diagnoser operations without providing specific algorithms. While in ours, since BPEL services are chosen as the application, local diagnosis algorithm is precisely defined. (Yan and Dague, 2007) has introduced an approach similar in its principle but different in its implementation: automata are used instead of Petri nets for modeling; trajectories in the synchronized product of the automaton model and the observations are used instead of data dependencies; the diagnostic algorithm is centralized instead of being decentralized.

REFERENCES

Ardissono, L., Console, L., Goy, A., Petrone, G., Picardi, C., Segnan, M., and Dupré, D. T. (2005). Enhancing web services with diagnostic capabilities. Proc. of European Conference on Web Services (ECOWS-05), pp. 182-191, Vaxjo, Sweden, IEEE.

Bauer, A. (2005). Simplifying diagnosis using lsat: a propositional approach to reasoning from first principles. In *vol 3524 of Lecture Notes in Computer Science*. Proc. of the 2005 International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR), Springer-Verlag.

Hamscher, W., Console, L., and de Kleer, J. (1992). *Readings in Model-based Diagnosis*. Morgan Kaufmann.

Li, Y., Melliti, T., and Dague, P. (2007). Modelling bpel web services for diagnosis: towards self-healing web services. Proc. of the 3rd International Conference on Web Information Systems and Technologies, pages 297-304, Barcelona, Spain.

Reiter, R. (1987). A theory of diagnosis from first principles. Artificial Intelligence, 32(1), 57-96.

Yan, Y. and Dague, P. (2007). Monitoring and diagnosing orchestrated web service processes. Proc. of the 2007 IEEE International Conference on Web Services.