BUSINESS PROCESS DISTRIBUTION AN INTELLIGENT APPROACH

[1,2]Faramarz Safi Esfahani, [3]Masrah Azrifah Azmi Murad, [4]Md.Nasir Sulaiman, [5]Nur Izura Udzir

[1]PhD Candidate., Department of Computer Science, University of Putra Malaysia, Malaysia.

[2] Department of Computer Science, Islamic Azad University, Najaf Abad Branch, Esfahan, Iran.

[3]Asstt Prof., Department of Computer Science, University of Putra Malaysia, Malaysia.

[4]Assoc. Prof., Department of Computer Science, University of Putra Malaysia, Malaysia.

[5] Asstt Prof., Department of Computer Science, University of Putra Malaysia, Malaysia.

Email: fsafi@iaun.ac.ir, {masrah, nasir,izura}@fsktm.upm.edu.my

ABSTRACT

Business Processes in Service Oriented Architecture (SOA) are run using an orchestrate engine. The point here is that running a huge number of business processes under a centralized orchestrate engine result in degrading of run-time environment abilities. Apart from this, running clustered orchestrate engines as an alternative way to obviate centralized orchestrate engine problems is not a final solution. On the other hand, there exist many researches focusing on decomposing or segmentation of business processes in run-time some of which attempts to decompose a business process to its building activities, while others break business process parts to sub flows or segments. Decomposing of a business process to its building activities will lead to a large number of activity agents in run time and it subsequently leads to more resource consumption and run-time system degradation. Segmentation, though, is useful however there are no criteria for business process segmentation commensurate with run-time environment requirements. In this paper, we introduce an intelligent process distribution method to first) increase business process adaptability with run-time environment, second) choose the best granularity for segments as well as encapsulating them in agents and third) decrease resource consumption due to reduced number of agents and messages. We also prove the correctness of our method mathematically.

Keywords: Adaptive systems, Business Process Mining, BPEL, Service Oriented Architecture, Mobile Agents, Workflow, Distributed Orchestrate Engine.

1. Introduction

In service oriented architecture, business processes are executed by an orchestrate engine that is responsible for running the activities of a process. Normally, a single engine is applied to manage a business process and scalability is satisfied by replicating orchestration engines which do not obviate the problems of centralized engines.

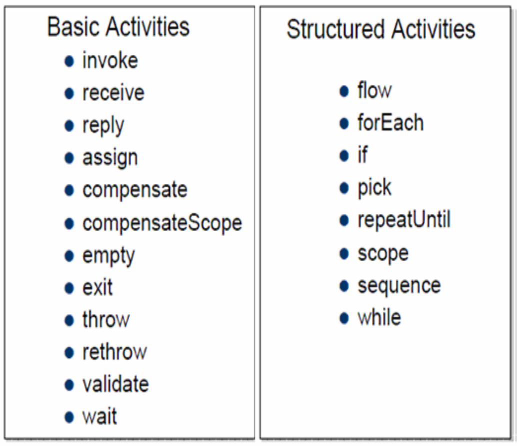
On one hand, many researchers are working on BPEL business process distribution. The main idea is distribution of activities of a BPEL process among some autonomous agents or sub processes interacting through a middleware. On the other hand, a number of researches have been focusing on the idea of process mining to extract useful information from process log files. The mined information will be used to detect most relevant parts of a business process, drawing run-time Petri net model of a business process and discovering social networks which are extremely important to provide more adaptable business processes with run-time environment.

In this paper we are going to reduce resource usage and improve the adaptability of business processes considering the execution history of previous executed processes. Accordingly, the contributions of this paper are: 1) Designing an intelligent method based on process mining to ameliorate the granularity of agents at compile time. 2) Improving the adaptability of the BPEL processes considering execution history of previous executed agents using process mining. 3) Reducing the number of agents and exchanged messages compared to the traditional methods of BPEL distribution.

2. Background and Related Work

BPEL: The Business Process Execution Language or BPEL briefly supports web services relationships and interactions in business transactions, message exchange correlation for long running message exchanges, parallel processing of activities, the mapping of data between partner interactions, consistent exception and recovery handling [1, 2].

Table 1: BPEL defines two types of activities



Basic Activities	Structured Activities
• invoke	• flow
• receive	• forEach
• reply	• if
• assign	• pick
• compensate	• repeatUntil
• compensateScope	• scope
• empty	• sequence
• exit	• while
• throw	
• rethrow	
• validate	
• wait	

According to table1, BPEL activities [1, 2] can be classified as basic activities that perform some primitive operations and structured activities that define control flow. The key BPEL basic activities are Invoke, Receive, Reply, Assign, Compensate, Compensate-Scope, Empty, Exit, Throw, Rethrow, Validate and Wait whereas structured BPEL activities are Flow, For-Each, If, Pick, Repeat-Until, Scope, Sequence and While, respectively.

Process Mining: Service Oriented Architecture contains a variety of events that can be logged. In addition, log data can be used for process mining purposes, its goal is to build models without apriori knowledge, based on sequences of events, one can look for the presence or absence of certain patterns and deduce some process models from it[3]. In [4] a framework for an agile mining of business processes introduced. In this framework, analyzing and mining of business processes change log information, the framework let process engineer to adapt business processes models based on the outcome of these analyses and to migrate related process instances to the new model. In this system, change log mining is being used to improve the adaptability of workflows while our IPD approach uses log mining to improve the granularity of agents in distributed workflows which will finally increases business process adaptability.

Web Service Interaction Mining (WSIM) [5] tries to mine log information provided by web service interactions. It also categorizes different levels of service mining based on log information. IPD is focused on the mining of business processes activities not interaction of web services only.

Using process mining to learn from process changes in evolutionary systems [6] based on the assumption that process changes are being recorded by system, two mining techniques have been offered to improve the adaptability of the process management system (PMS). The found changes during process mining provide an overview of the changes happened until now. In [7] a process mining for change logs proposed to apply. They not only analyze the operational processes but also the adaptations made at the process type or process instance level. They provide a Petri net model of the changes using process mining to find the most changed parts of a process. Our work also mines process log information (not change log information) to detect most relevant activities of a process using a process mining method and based on the achieved results we will distribute business processes to improve quality factors.

BPEL Decomposition and Interaction Middleware: In Publish/Subscribe [2, 8, 9] communication, the interaction between the information producer (publisher) and consumer (subscriber) is mediated by a set of brokers. Publishers publish events to the broker network, and subscribers subscribe to interesting events by submitting subscriptions to the broker network. It is the responsibility of the brokers to route each

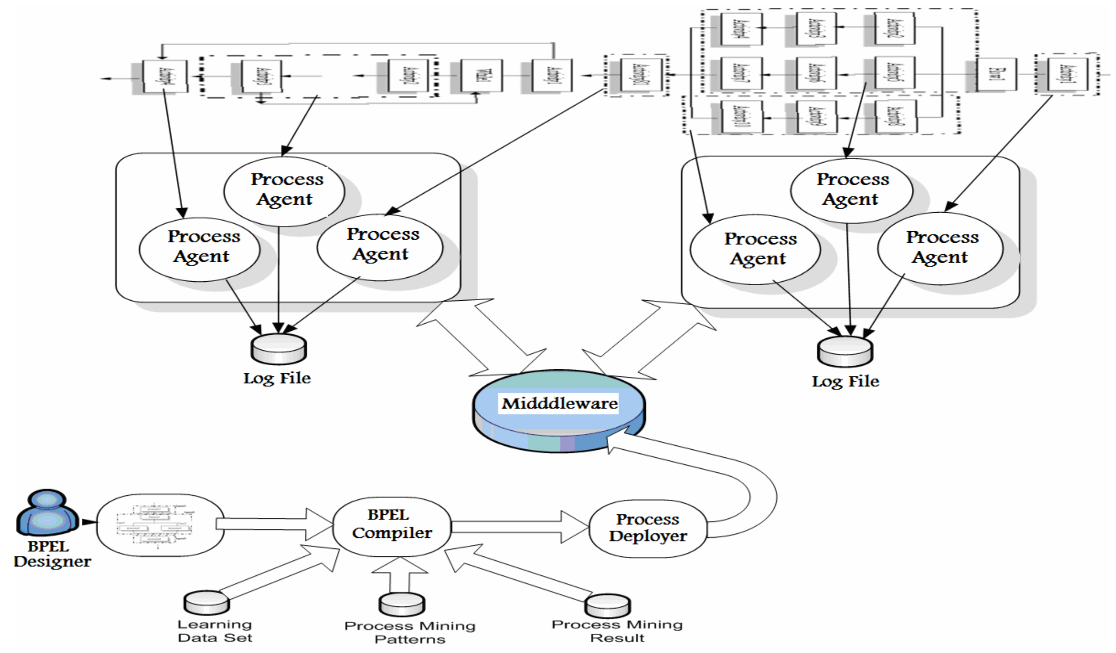


Figure1: System Architecture

event to interested subscribers. NINOS[2] uses a Publish/Subscribe messaging service to handle interaction of agents. In addition, another well known method of agent communication is using the concept of Tuple Space implemented in LINDA language. ReSpecT [10] uses an extension of Tuple Space idea in LINDA platform called ReSpecT to realize the cooperation of agents. Furthermore SOA stack supports messaging and [11] uses SOA messaging protocols and WSDL to wire decomposed components.

In NINOS[2], a distributed agent-based orchestration engine presented in which several agents execute a portion of a business process and collaborate in order to execute the whole process. Similarly, ReSpecT [10] uses the same idea to distribute a workflow. In addition, [11] presents a mechanism to partition a business process so that each partition can be enacted by a different participant. In fact [11] disconnects the partitioning itself from the design of the business process. All these methods [2, 10, 11] do not have any control on the number of produced agents, granularity as well as adaptation of agents with the run-time environment. While IPD uses a mining process method to discover the useful patterns to provide suitable agents.

These methods [2, 10, 11] are most relevant works to our approach from BPEL distribution and decomposition point of view. IPD is not depended to the method of communication and wiring of partitions or agents. It is worth mentioning that our model is independent of messaging middleware therefore, agents in our model can use any other types of middleware to interaction

3. Intelligent Process Distribution (IPD)

In this part, we discuss about the architecture of IPD system. The main purpose is the intelligent distribution of BPEL processes using a mining process approach. As mentioned before, in traditional methods a process is decomposed to its ingredients (or activities) and each activity is encapsulated in an agent. The huge number of produced agents, their resource usage, high number of agent interactions and exchanged messages are the result of decomposing a process to its activities in the lowest granularity. In the proposed method we will use mining patterns to detect most relevant, closely related as well as juxtaposed activities from a spatial view. Then we encapsulate closely related activities in agents and their interactions are as [2, 10]. IPD will produce agents which are in their most suitable granularity. It results in more adaptable agents, in run-time environment, through a reduced number of produced agents and less message passing.

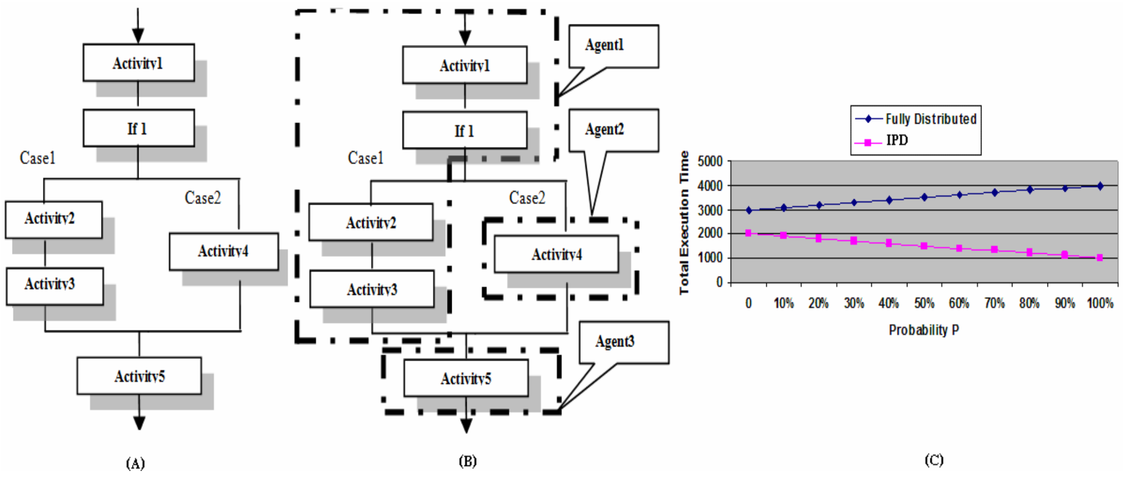


Figure 2: Using If Activity in Two Methods

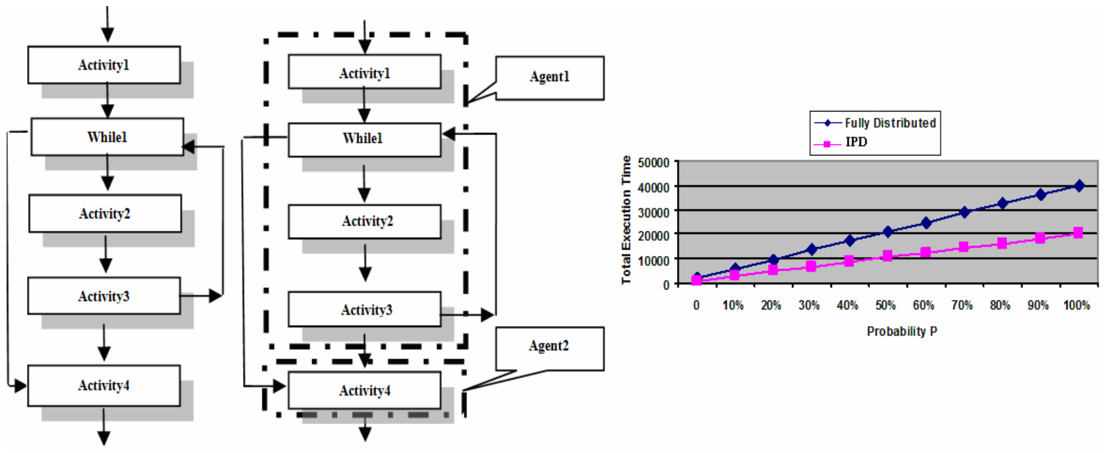


Figure 3: Using While Activity in two Methods

3.1 The IPD System Architecture

The recommended system architecture is depicted in figure1. According to IPD architecture, a **process agent** is an agent contains most relevant activities encapsulated in one agent. The **distributed process log files** are based on the number of nodes involved in executing agents. As the number of nodes containing process agents increases, the distribution of log files will be increased. **BPEL Compiler** is supposed to be a traditional BPEL compiler but it has to be equipped to convert a BPEL process to some agents according to agent detection patterns which will be discussed in the following sections. **Middleware** is a broker through which agents can exchange messages. **Learning dataset** is used for starting up a process, when mining result set is empty. Developers can initialize a process using learning data set to ameliorate system initialization. Being warmed up the system, the process compiler will use fresh mining data from process mining result set. **Process mining result set** is actually a repository to maintain mining patterns extracted from log files.

3.2 The Structure of Log Files

To do a comprehensive mining the following log file structure is considered. Each process agent is responsible for producing the log file of its activities. We consider six fields as entries of process log files as follows:

ProcessTypleId shows the identification number of a specific BPEL process. The next field is ProcessVersionId that shows the version of a compiled process. This field is due to the

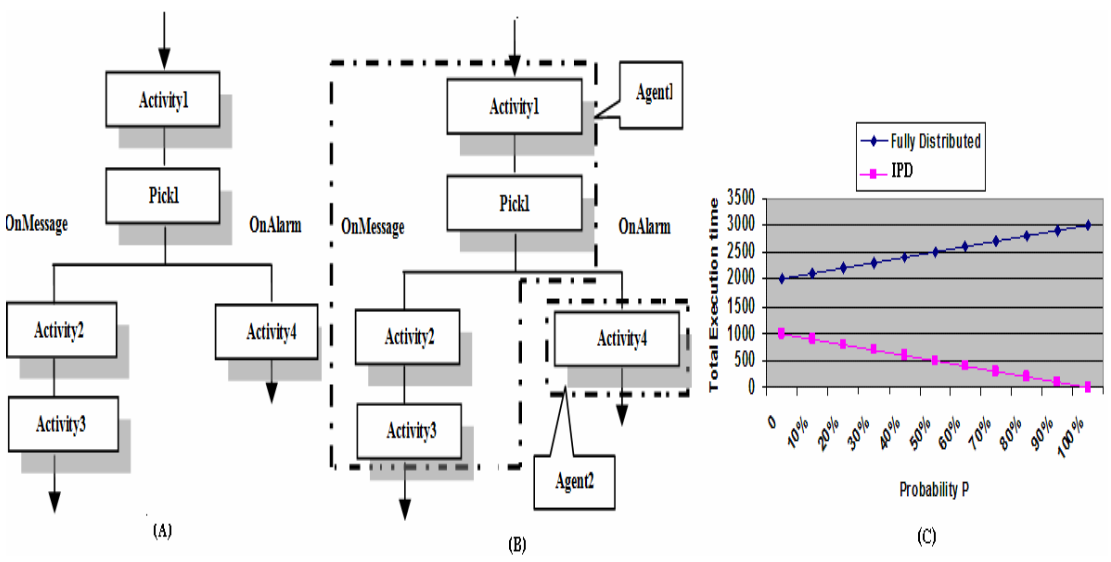


Figure 4: Using Pick Activity in Two methods

fact that there will be different compilations for a specific BPEL process. Furthermore, InstanceId shows the exclusive number of a process instance. In addition, ActivityId demonstrates the identification number of the running activity. The NextActivityId field shows the next activity selected to be run and has to be determined at runtime as well. There might be more than one activity to be run after finishing up one activity. Finally, the ElapsedTime specifies the total amount of time elapsed from an activity execution start time and normally, it can be further divided to start time and finish time subfields. Agent Detection Patterns

Owing to the fact that a number of activities in a process are juxtaposed and closely related, there is a tendency in activities of a process to call their spatial neighbors. Hereafter, we examine the BPEL activities one by one to show how process mining playing role is. We will prove our method for more complex BPEL activities from a mathematical point of view as well. For more simplicity, we consider equal execution time (t_A) and communication time (t_C) for all activities. In addition, we believe that the time cost of message passing is much more than the execution time of one activity $(t_C \rangle\rangle t_A)$ except those activities like invoke, receive or reply sending or receiving message to external web services. Indeed, we presume $t_C \approx 1000 t_A \wedge t_A = 1$ everywhere the numerical value of (t_A) and (t_C) are needed, based on an empirical experiment.

3.2.1 If (Switch) Activity

The If (Switch) activity is a structured activity and is built using an ifelseif-else format. According to [1] the If activity lets you choose exactly one execution path from among many such as If structure in common programming languages.

According to [2, 10], the sample BPEL process in figure2(a) should be converted to six agents. The execution time of the above If activity with probability of p for executing Case1 and probability of 1-p to execute Case2 is as follows:

$$t_{Overall} = t_{A1} + t_C + t_{if} + p(t_C + t_{A2+}t_C + t_{A3}) +$$
$$(1-p)(t_C + t_{A4}) + t_C + t_{A5} \approx 4t_A + 3t_C + p(t_A + t_C)$$

Suppose that the recommended process mining method detects the execution path including Activity1, If1, Activity2 and Activity3 is more traversed than the alternative path including Activity3. Therefore three agents should be appeared according to figure2(b) The total time for processing If activity is as follows:

$$t_{total} = p\times(t_{A1}+t_{if1}+t_{A2}+t_{A3})+(1-p)(t_{A1}+t_{if1}+t_C+t_{A4})+t_C+t_{A5}$$
$$\approx p\times 4t_A+(1-p)(3t_A+t_C)+t_A+t_C$$

Obviously, increasing the number of activities increases the communication time in fully distributed approach. Also, it is process mining algorithm that detects the probability of p is high which results in reducing of t_{total}. Figure2(c) compares the behavior of two approaches.

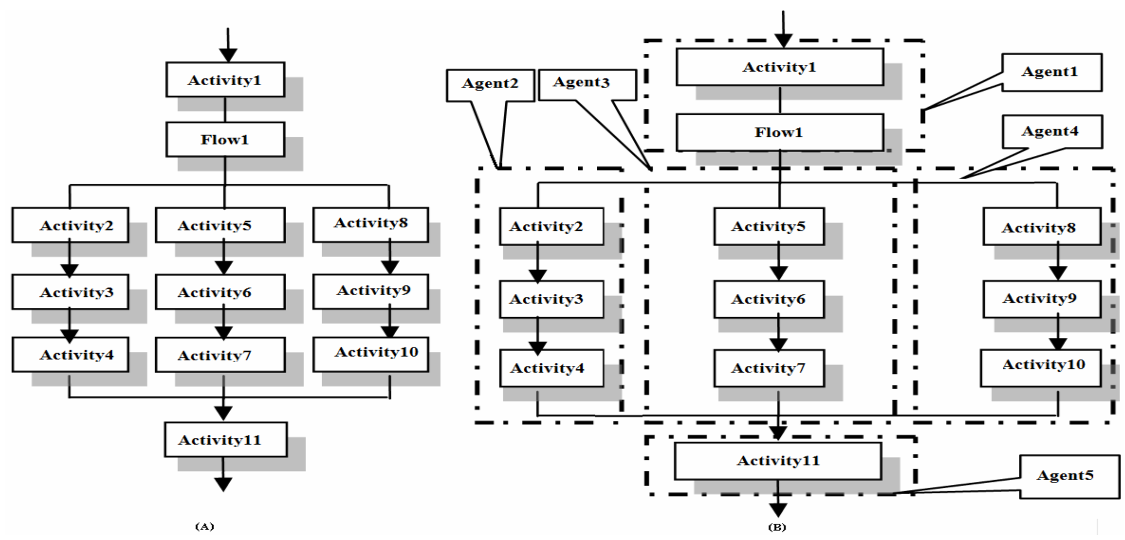


Figure 5: Using Flow Activity in Two Methods

3.2.2 While Activity

In BPEL specification, we have three loop structures among which we start with While loop. The While repeats the enclosed statement block until the conditional statement evaluates to False[1].

A sample While activity is shown in figure3(a). In [2, 10] the While activity is converted to a While agent. Evaluating the condition, the While agent triggers the subsequent activity. We assume that the possibility of repeating the loop for n times is p and with possibility of 1-p the loop condition will not be satisfied. The total execution time for While is as follows:

$$t_{total}=t_{A1}+t_C+p\times n\times(t_w+t_C+t_{A2}+t_C+t_{A3}+t_C)+(1-p)(t_w+t_C)+t_{A4}\approx 2t_A+t_C+p\times n\times(3t_A+3t_C)+(1-p)(t_A+t_C)$$

Again using IDP the activities in the While structure will be encapsulated in three agents owing to the fact that they are closely related as it is shown in figure3(b). So, the total execution time using IPD will be calculated as follows:

$$t_{total}=p\times n\times(t_{A1}+t_W+t_{A2}+t_{A3})+(1-p)(t_{A1}+t_W)+t_C+t_{A4}\approx p\times n\times(4t_A)+(1-p)(2t_A)+t_A+t_C$$

Based on the assumption that process mining has already detected that the probability of p is high therefore, we expect better result comparing with the fully distribution model. For this While activity we consider the number of loop iterations is n=100. Obviously, the output result is highly depended on the value of n. Figure3(c) shows the comparison charts of two methods.

3.2.3 Pick Activity

The Pick activity forces the process to wait until one event is triggered. All of these events are either onMessage or onAlarm elements. You can have as many onMessage and onAlarm activities as you want, but exactly one of them will be executed. Once one event is executed, all others are disabled[1].

Figure4(a) shows a sample Pick activity including two onMessage and onAlarm elements. In its normal distribution according to [2, 10] it should be converted to four activities. We assume that the probability of running onMessage is p and in other cases OnAlarm would be run. So, the total time for running this activity is as follows:

$$t_{total}=t_{A1}+t_C+t_P+p(t_C+t_{A2}+t_C+t_{A3})+(1-p)(t_C+t_{A4})\approx 3t_A+2t_C+p(t_A+t_C)$$

Suppose that mining process approach detects Activity1, Pick1, Activity2 and Activity3 are highly relevant through OnMessage element. The BPEL compilation is in figure4(b). The total time to execute this Pick activity is as follows:

$$t_{total}=p(t_{A1}+t_P+t_{A2}+t_{A3})+(1-p)(t_{A1}+t_p+t_C+t_{A4})\approx p\times 4t_A+(1-p)(3t_A+t_C)$$

Obviously, increasing the number of activities increases the communication time in fully distributed approach. Considering the hypothesis

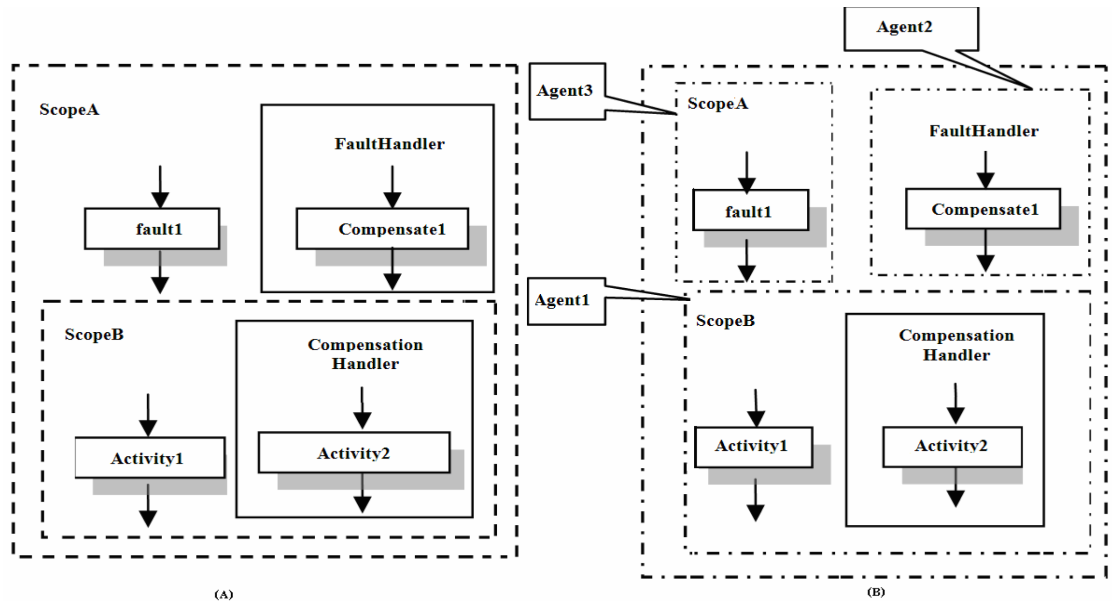


Figure 6: Scope, Compensate and Fault

that process mining has already detected the high probability of p, therefore, decreasing total execution time is expected. Also, Figure4(c) shows the comparison of two mentioned methods for Pick activity.

3.2.4 Flow Activity

A Flow activity defines one or more child activities that execute concurrently, which is the most basic use of this construct. The Flow activity also allows us to synchronize activities, such that one activity starts when another ends. We've dealt with activities that implement *program-like* structures until now, but with Flow we can do *graph-like* structures[1].

Figure5(a) shows a sample Flow activity in its simplest use. Accordingly, there are three branches in the Flow structure which are able to run concurrently. Based on what is recommended in [2, 10] this sample structure will be compiled to twelve agents. The total time for processing this Flow activity is as follows:

$$t_{total}=t_{A1}+t_C+t_F+t_C+\max[(t_{A2}+t_C+t_{A3}+t_C+t_{A4})\vee (t_{A5}+t_C+t_{A6}+t_C+t_{A7})\vee(t_{A8}+t_C+t_{A9}+t_C+t_{A10})]+t_C+t_{A11}$$

$$t_{total}\approx 6t_A+5t_C$$

In contrast, applying IPD method results in three groups of closely relevant activities. Indeed, the normal way of handling this situation is creating one agent for each group and the total number of produced agents will be as in figure5(b). The total time to execute this Flow activity using IPD would be as follows:

$$t_{total}=t_{A1}+t_F+t_C+\max[(t_{A2}+t_{A3}+t_{A4})\vee(t_{A5}+t_{A6}+t_{A7})\vee(t_{A8}+t_{A9}+t_{A10})]+t_C+t_{A11}\approx 6t_A+2t_C$$

We come to the conclusion that our method improves Flow activity total execution time according to $6t_A+2t_C \langle\langle 6t_A+5t_C$. Obviously, increasing the number of activities increases the communication time in fully distributed approach.

3.2.5 Scope, Compensate and Fault Activities

Scopes allow us to break up our business processes into logical units of work. In fact, Scopes provide a context for the execution and/or documentation of enclosed activities, and they can have variables that are visible and usable within the Scope level. Scopes can have both default and defined Fault and Event handling logic, and they can be undone, if necessary. Also, undoing the work of a Scope involves the concept of compensation. When designing your BPEL processes they should be organized into logical units of work that can be undone[1].

From Compensate point of view, previous activities which had successfully completed may need to be undone. In addition, due to the nature of business process, it is usually long-running and asynchronous. In BPEL, a Compensation

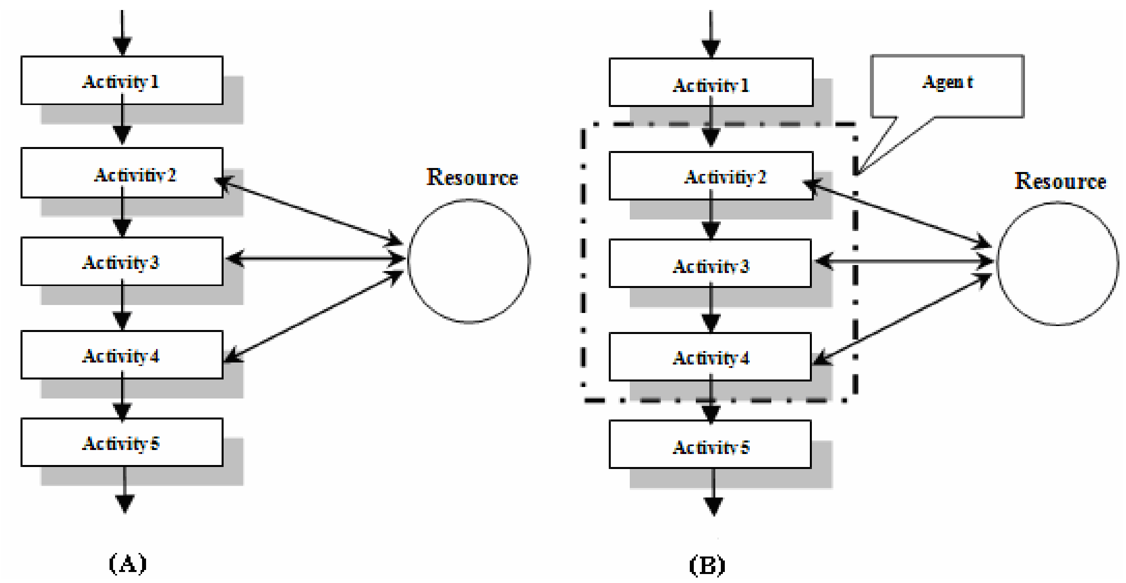


Figure 7: Closely Related Resources Detection

Handler is a local declaration, done at the Scope level only, and is not normally available for the Process itself.

However, designers add extensions to allow process level compensation handling. In contrast, Fault Handlers enable the process to recover from abnormally terminated actions, whereas Compensation Handlers undo successfully completed actions. The other difference is the context from which they are called. A BPEL Fault Handler is invoked in response to a Fault in the same context (i.e., the same Scope), whereas Compensation Handlers are invoked in response to a Fault in a higher context (i.e., the parent Scope). Moreover, the order in which compensation handlers are performed usually makes a difference. By default, compensation is performed in the reverse order of the completion of Scopes involved[1].

According to [2, 10] a Scope activity will be converted to an agent and the agent will be in contact with its included activities and Fault and Compensation handlers through a Pub/Sub or a tuple space middleware. From IPD point of view, Scope activity might be considered as a separate activity or might be encapsulated with its Compensate and Fault handlers.

According to NINOS[2], a simple Compensate as shown in figure 6(a), will be converted to two Scope agents A and B. Scope A's Fault handler involves the compensation handler in Scope B. The Scope agent for Scope B subscribes in compensation events for the Scope and triggers the first activity in its compensation handler using a publication method. So, Scopes as well as Compensates have been converted to agents while using Publish/Subscribe messages to interaction. The same behavior has been done using tuplespace in [10].

If we presume that the output of the mining algorithm shows Compensation Handler in Scope B is frequently being used then it will be possible to encapsulate Activity 1 along with Activity 2 and Scope agent B all together to reduce the number of agents. The produced agents using our method have been drawn in figure 6(b) for more illustration.

In this case putting together those Fault or Compensate handlers which are closely related in one agent will omit communication costs of an agent interaction which results in decreasing total execution time of Scope activities and its included Compensate and Fault handlers.

3.2.6 Sequence Activity

A Sequence is a structured activity which can contain other activities. The purpose of a Sequence is to define the execution order for a group of activities. Sequences, though can contain other Sequences and can be nested as deeply as you want. Sequences have all the standard attributes and elements and they must contain at least one or more activities[1].

According to NINOS [2] all of the activities in sequence should be considered as a separate activity. Sequence activity from our point of view will be relatively straightforward due to its simple structure. The total time to execute a sequence of n activities is expected to be as $n \times t_A + (n-1) \times t_C$.

When there is a sequence of activities running sequentially, according to an Apriori mining they will be detected as closely related activities and will be encapsulated in one agent, therefore, the total time using our method will be $n \times t_A$ which is much less than $n \times t_A + (n-1) \times t_C$ in fully distributed method.

3.2.7 Closely Related Resource Pattern and Other BPEL Activities

It is worth mentioning that the high interaction of one or more activities with one resource makes them closely related. So, encapsulating of these activities would be possible. It results in less communication cost and we will be able to put the produced agent to a suitable location where the cost of interaction with resource is lessened. Figure 7.a shows an interaction between activities and a resource.

If the mining algorithm comes up with the surmise that Activity1, Activity2 and Activity 3 are closely related not only with each other but also with the resource, as depicted in figure16, and encapsulating these activities together as an agent and locating the agent in a right place would mitigate the communication costs with the resource, therefore it would be obvious that mining algorithm will do encapsulation as shown in figure7.b.

3.2.8 Other BPEL Activities and Nested Cases

Other BPEL activities are into a large extent straightforward and need less study comparing with mentioned activities. Also, in addition to the discussed basic cases, it is worth mentioning that more complex business processes and nested ones are composed of simple cases and demand further attention as future work.

4. Conclusion

In this paper, we present the concept of Intelligent Process Distribution or IPD using mining idea to increase business process adaptability and decrease resource usage. Our contributions in this paper are: firstly, we propose the idea of process distribution using process mining. Secondly, we illustrate several mining patterns for some paramount BPEL activities to show how agent producing is based on the execution history of previous executed business processes. We also prove the method using a mathematical approach. By and large, our idea will results in distributing a business process to some agents which are in their best granularity, neither fully distributed nor fully centralized, actually based on the run-time behavior of previous executed business processes. IPD on one hand increases the adaptability of business processes with run-time environment and on the other hand decreases the number produced agents as well as the number of messages for agent interactions.

At present time we are developing a methodology and implementing the idea of IPD as well as comparing IPD with other mining algorithms. For future work, we would like to extend our work by designing a mining algorithm based on temporal invocation of activities in a process. In addition, designing a distributed mining algorithm would be of our future plan. Providing a plug-in for ProM IDE [12] tool of process mining is another recommendation as well.

5. References

[1]Active-Endpoints, "ActiveBPEL Engine - Open Source BPEL Server," 2008.
[2]V. M. Guoli Li, and Hans-Arno Jacobsen, "NiNos: A distributed service oriented architecture for business process execution," *Technical report, Middleware Systems Research Group*, July 2007.
[3]"Process Mining and Monitoring Processes and Services: Workshop Report," in *The Role of Business Process in Service Oriented Archtictures*, Eindhoven University of Technology, P.O.Box 513, NL-5600 MB, Eindhoven, The Netherlands., 2006.
[4]B. Weber, M. Reichert, S. Rinderle, and W. Wild, *Towards a Framework for the Agile Mining of business Processes* vol. 3812/2006: Springer Berlin / Heidelberg, 2006.

[5]R. Gombotz and S. Dustdar, "On Web Services Workflow Mining," in *Springer-Verlag*, B. e. al, Ed. Berlin Heidelberg, 2006, pp. 216-228.

[6]S. R.-M. Christian W. Gunther, Manfred Reichert, Wil M.P. Van Der Aalst, Jan Recker, "Using process mining to learn from process changes in evolutionary systems," *International Journal of Business Process Integration and Management* vol. 3, pp. 61-78, 2008.

[7]W. M. a. G. van der Aalst, Christian and Recker, Jan C. and Reichert, Manfred, "Using Process Mining to Analyze and Improve Process Flexibility - Position Paper," in *Proceedings 18th International Conference on Advanced Information Systems Engineering, Proceedings of Workshops and Doctoral Consortium*, 2006, pp. 168-177.

[8]H.-A. J. F. Fabret, et al, "Filtering algorithms and implementation for very fast publish/subscribe systems," *In ACM SIGMOD*, 2001.

[9]D. S. R. A. Carzaniga, and A. L. Wolf., "Design and evaluation of a wide-area event notification service," *ACM ToCS*, vol. 19(3):, pp. 332–383, Aug. 2001.

[10]E. D. Mirkov Viroli, Alessandro Ricci, "Engineering a BPEL orchestration engine as a multi-agent system," *Elsevier*, January 2007.

[11]F. L. Rania Khalaf, "E Role-based Decomposition of Businesses using BPEL," in *IEEE International Conference on Web Services(ICWS'06)*, 2006.

[12]"ProM IDE, The Process Mining Group, TM.IS department, Eindhoven University of Technology.," 2007.