Dynamic Adaptation of Business Process Based on Context Changes: A Rule-Oriented Approach

Guangchang Hu, Budan Wu, and Junliang Chen

State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876
{hgc,wubudan,chjl}@bupt.edu.cn

Abstract. In a dynamic environment, business process needs to be adjusted and evolved in response to the changeable internal policies and external environment. However, it is a time-consuming and laborious way by redesigning process model and executing the process instance. In this paper, we propose a rule-oriented approach to dynamically generate business process according to the current context at runtime. To enable dynamic and context-aware adaptation, the relationship between services and context is described as rules, which are then used to generate the solution with a mapping mechanism. Two algorithms are designed to generate the activity sequence at runtime, which is the solution of process adaptation. In order to achieve the preference selection, a process assessment strategy has been proposed to constrain the generated activity sequence. Simulation experiments have been conducted to demonstrate the efficiency of our approach.

Keywords: BPM, adaptation, dynamic assembly, service composition, context-aware.

1 Introduction

Service-oriented computing is a new computing paradigm. There are many advantages to organize business processes in the form of services [10]. A well designed business process can increase effectiveness and add value for the enterprise. However, the ability to respond to the changes in business processes is very small. The changes come from the adjustment of internal regulatory policy, as well as the impact of environment context change [5]. Due to the changes, a predefined business process may become unable to continue or no longer meeting the current business goals. So, business process needs to be adjusted and evolved in response to the changeable internal policies and external environment [14]. However, it will be a time-consuming and laborious work by redesigning process model and executing the process instance. Therefore, how to rapidly adjust the predefined business processes to meet business goals at runtime according to the current context is a meaningful problem. Especially in Internet of Things (IoT), the dynamic characteristics of the environment and the burstiness of services will directly affect the running process instances. It has practical significance to reassemble business activities to adapt to the changes by the way of dynamic planning at runtime.

A.R. Lomuscio et al. (Eds.): ICSOC 2013 Workshops, LNCS 8377, pp. 492–504, 2014.

The traditional workflow mainly concern to design business process model by analyzing business requirements at design stage, and execute the business process instances at runtime [18]. Both the definition of process logic and the binding of services can be realized at design stage. This kind of workflow is suitable for fixed business, and it is a static process, e.g. alarm process using WS-BPEL. But in a dynamic environment, the changing context or abnormal events will lead to the initial process instances cannot continue. And the changes of application requirements will lead to the initial business processes cannot meet the new business goals. Some changes are unpredictable, and it may only occur once, or only reappear under the certain situations. So, it is a time-consuming and laborious way to adapt to the changes by redefining the process models back to the design stage. And it is a good solution to generate a temporary process which meets business goals and user preferences by a rule-oriented approach according to the current context and various rules at runtime.

2 Related Work

The process is too rigid is the main problem in traditional workflow management systems (WFMSs). These systems are not suitable for handling rapidly evolving processes [7]. The case-handling paradigm, such as FLOWer, is usually considered as a much more flexible approach, which allowing users to modify the predefined model [1]. A new generation of adaptive workflow management systems are developed [8,19], such as ADEPT, which response to the demand of dynamic business process management. However, both in case-handling and adaptive systems, process models are presented in a process modeling language (e.g., Pi calculus, Petri nets [13], etc.), which precisely prescribes the algorithm to be executed. Although case-handling and adaptive systems allow for changes of models written in imperative languages, the result remains an imperative model. This can result in many efforts to implement various changes over and over again. In order to avoid the mandatory features of these modeling languages, Pesic et al. [11] proposes ConDec as a declarative language for modeling business process. Unlike imperative languages, declarative languages specify what not to do instead of specifying how to work. This leaves a lot of room for the maneuver of users who can make decisions and work in various ways with the same ConDec model. But this method strongly depends on the description of constraint. The flexibility is very weak when the description is insufficient [15]. And it cannot access the runtime state of the process [2].

There already exist many researches on the adaptability of business process in the field of business process management [4,12,17]. Bucchiarone et al. [3] defines abstract activities of process model according to business goals at design stage. The proposed method is fragment-based, and it gradually refines the implementation of abstract activities by a series of adaptation mechanisms at runtime. However, the adaptation mechanisms are relatively complex, and it cannot support manual intervention and adjustment for the generated business process. Moreover, this method does not support to user preferences. Yu et al. [20] uses the technology of aspect-oriented. And the proposed method can realize different user's preferences which are weaved in processes in the form of aspect. But the process logic of process model is fixed, and its variability only reflects in the replacement of a specific service according to different user at

runtime. Mejia et al. [9] maps the business processes to a series of ECA rules. And the proposed method can realize process reengineering according to business requirements. However, this method is more dependent on the business experts. Due to services and process logic are bound in ECA rules, it has a lower flexibility to cope with the changes in the changeable environment.

Our research is based on the previously developed information service platform of heating system. The platform is an IoT application project about urban central heating and room temperature monitoring system. The architecture of this platform is shown as Figure 1. This paper is research on the adaptive module (the dashed rectangle of Fig.1) for BPM. We encountered a dynamic assembly problem between main process and sub-process at runtime, when we developed the application of the Maintenance Process with this platform. Based on this problem, we propose a rule-oriented dynamic planning approach (called RoDP) which can quickly respond to the changes at runtime. The RoDP approach can realize dynamic assembly and adaptive adjustment of the interrupted business process at runtime.

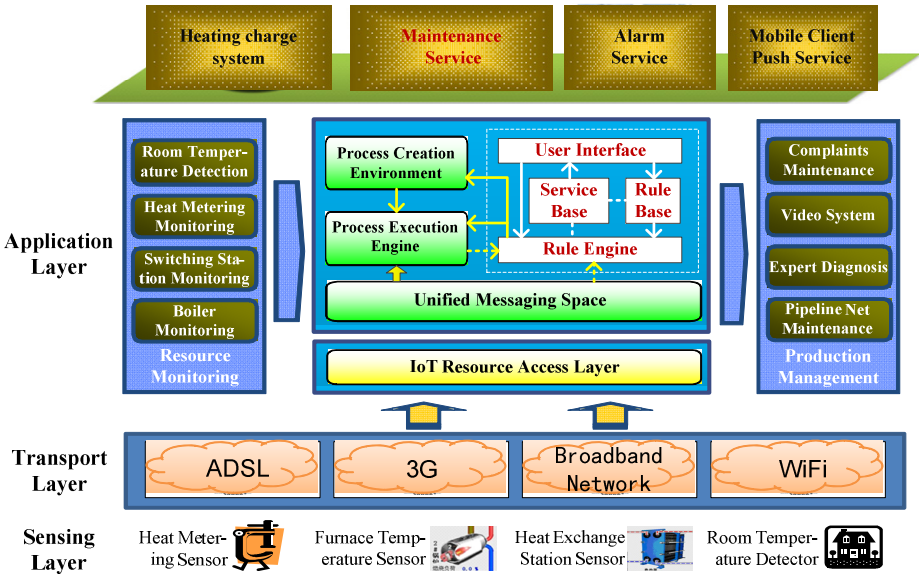

Fig. 1. Architecture of information service platform

3 Scenario and Problem Description

The following scenario is about heating maintenance process of the Maintenance Service. There are three ways to submit a maintenance request. When a request is submitted, duty manager assigns repairman to repair and confirms the result of the maintenance. The maintenance process is shown as Figure 2.

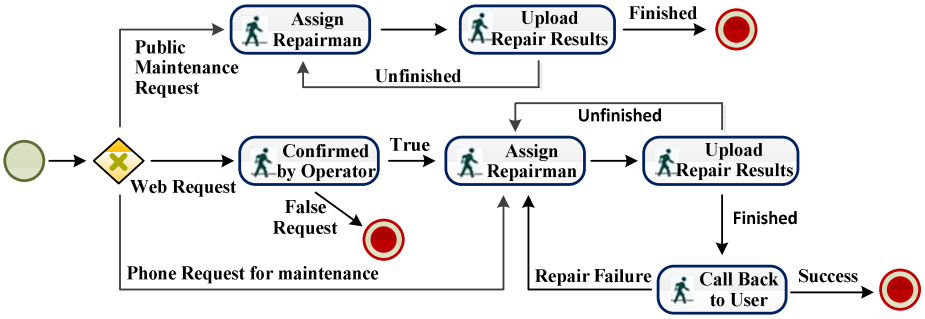

Fig. 2. Heating maintenance process of Maintenance Service

The changes of context information may interrupt an ongoing maintenance process instance. For example, if there is a lack of some materials during the maintenance period, the process instance will be unable to continue until the materials are supplemented. So, the main process instances need to dynamically assemble and execute the sub-process of procurement, and return to the process instances to continue until the total business goals are achieved. In addition, the heating maintenance process may not be suitable for some new requirements. For example, heating equipments or temperature sensors need to be replaced uniformly. So, a new process needs to be dynamically generated to achieve this new goal.

Therefore, one of the problems is how to dynamically assemble the relevant sub-processes and return to the main process when the execution of process instance is interrupted by some events. Another problem is how to dynamically generate the relevant processes to meet the new goals according to the current state, when new business requirements appear. Figure 3 shows the procurement sub-process about the event of lack of material. Figure 4 shows the new process about the new requirement of the replacement of heating equipments or temperature sensors. In conclusion, the general problem is how to rapidly organize business process to meet business goals at runtime according to the current context and the existing service, when the changes of contextual information or the emergence of new requirements affect the execution of process instances in the dynamic environment. The key to solve the general problem is to generate a related activity sequence to meet the business goals.

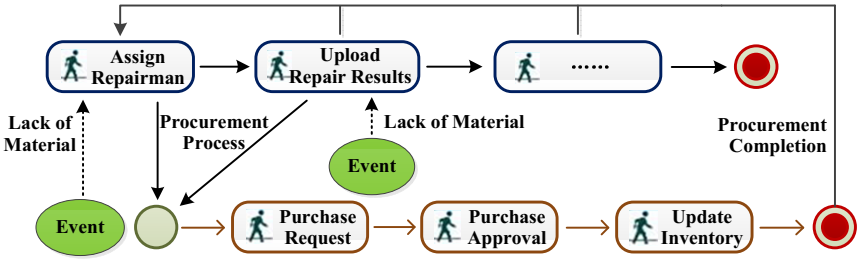

Fig. 3. Process of the procurement sub-process

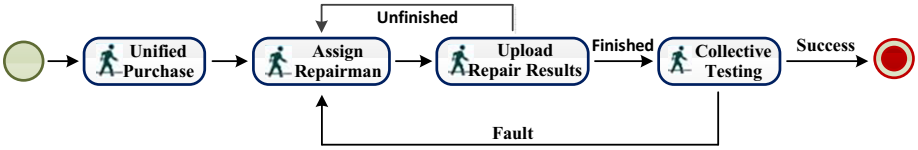

Fig. 4. Process of the replacement of heating equipments or temperature sensors

4 The RoDP Approach

In this section we propose our RoDP approach and discuss several key principles used in the design of this approach. First, we describe the adaptation architecture and the function of each component. Second, we introduce formal definitions of the elements of our adaptation architecture. Finally, we propose two algorithms to generate the activity sequence at runtime, and we conduct experiments and comparative analysis. In addition, we describe the mapping mechanism and assessment strategy in detail.

4.1 The RoDP Architecture

The adaptation architecture is shown as Figure 5. The predefined business processes are stored in the Process Model which is a part of the Process Creation Environment. Process instances are executed in the Process Execution Engine and receive the events from the Unified Messaging Space. In response to these events and new requirements, Rule Engine generates the corresponding activity sequence according to various rules which describe the relationships between states of resources and activities. Rules are stored in the Rule Base, and activities are stored in the Service Base. The relationship between services and rules is realized by the Mapping module in the User Interface. Different user preferences are realized by the module of Process Assessment.

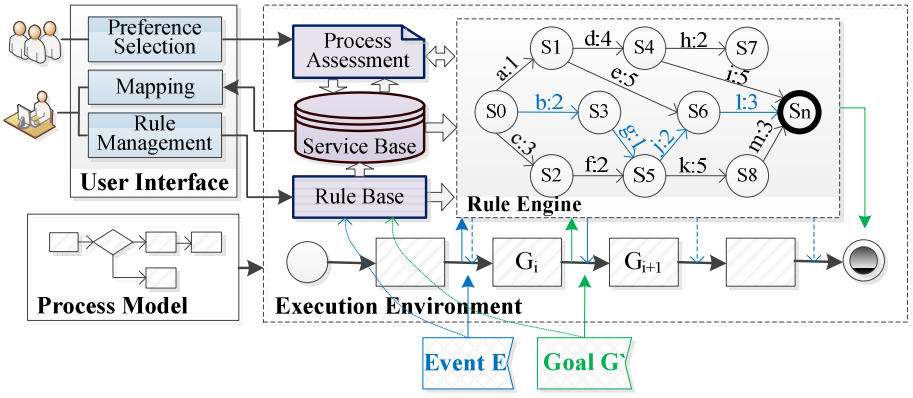

Fig. 5. The architecture of adaptive process

When the process instances are interrupted by events or new requirements, Execution Engine extracts the current states of the related resources and the objective states of the remaining activities. According to the current states, the objective states and the relevant rules, Rule Engine generates a weighted directed graph (called WDG), and then outputs the corresponding activity sequence according to assessment strategy. In addition, for the interruption caused by events, Execution Engine just return to process breakpoint or anywhere of the remaining activities in process model.

4.2 Definition

In this section we introduce formal definitions of the elements of our adaptation architecture.

1) *Context:* Context describes the current states of the resources. Many things can be seen as resources especially in IoT, such as sensor, material, person and other physical resources. Virtual resource can also be seen as an operable resource, such as data, Web service. Each resource has two or more states to mark the current feature of the resource. For example, the state of the material *I* is none (or yes), and the state of the Web service *J* is unavailable (or available).

Definition 1: *(Context)* $S_o=<\mathbb{E}_{id}(i)|\mathbb{E}_{id}(i).State, i\in N>$, where: S_o represents the current states of related resources of the interrupted instances; $\mathbb{E}_{id}(i)$ is the serial number of resource *i*; $\mathbb{E}_{id}(i).State$ is the current state of the $\mathbb{E}_{id}(i)$ resource and *State(a)* represents the resource state, and $0\le a\le\varepsilon$; ε is the quantity of the states of a specific resource.

2) *Activity*: Activity (or service) describes the input, output, a set of operations and the corresponding weight. In addition, the data which transmit between activities can be defined and accessed in the form of Artifact [16].

Definition 3: *(Activity)* $A=<\mathbb{A}_{id}, \mathbb{E}_{id}, Input, Output, Weight>$, where: $\mathbb{A}_{id}$ is the activity identity, and $\mathbb{A}_{id}(x)$ is the serial number of activity *x* ($0\le x\le\mu$), μ is the total number of activities; *Input* represents the triggered state of activity or input parameter of Web service, and *Output* represents the changed state or output parameter; *Weight* represents the cost, response time, user expectation or other QoS parameters of the activity; the $\mathbb{A}_{id}$ and $\mathbb{E}_{id}$ of virtual resource is the same, and the $\mathbb{A}_{id}$ and $\mathbb{E}_{id}$ of physical resource is different.

3) *Business Goal:* Business goal describes the states, which the remaining activities of process model should achieve. Business goal is generated by extracting all the outputs of unfinished activities when the process instance is interrupted by events. And business goal is *G`* which represents the final states when a new business requirement *G`* appears.

Definition 2: *(Business Goal)* $G=<S_x, S_{x+1},\ldots, S_n>$, where: $0<x<n$; and S_x represents the objective state which the first unfinished activity *x* should achieve, and $S_x:=\mathbb{A}_{id}(x).State(b)$; S_{x+1} represents the objective state which the next activity *x+1* should achieve in the process model; S_n represents the final state which the last activity should achieve.

4) *Rule*: Rule is an abstract description of the activity, which describes the precondition, result and operation for performing an activity.

Definition 4: *(Rule)* $R=<\mathbb{R}_{id}, A_{id}, precondition, result>$, where: $\mathbb{R}_{id}$ is the rule identity, and $\mathbb{R}_{id}(\alpha)$ is the serial number of rule α ($0\leq\alpha\leq m$), m is the total number of rules ; *precondition* is the condition that an activity is triggered; *result* is the result of implementation of the activity;

5) *Activity Sequence:* Activity sequence describes the ordinal relation of activities. According to the Process Assessment strategy, the activity sequence is generated by the RoDP algorithm and returned by the Rule Engine.

Definition 5: *(Activity Sequence)* $P=<A_{id}(x), A_{id}(y),\ldots, A_{id}(z)>$.

4.3 The Rule Mapping Mechanism

Rule is an abstract description about the activities of the corresponding physical resource and services of the corresponding virtual resource. The state of a physical resource can usually be changed by an atomic activity which has the same input and output. For example, the procurement activity changes the state of material *I* from none to yes, and the input and output parameters are all about the change of material quantity. So, the precondition and result of the rule respectively are the states (before and after the change) of the resource. The state of a virtual resource represents that the corresponding service is available or not. And the precondition and result of the rule respectively are the input and output parameter of the service. If an activity or a service realizes the transition from one state to another, it can be expressed as follows:

Physical resource	Virtual resource
$A_{id}(x)::E_{id}(i).State(a)\rightarrow E_{id}(i).State(b)$	$A_{id}(y)::E_{id}(y).State(a)\rightarrow E_{id}(y).State(b)$
$\mathbb{R}_{id}(\alpha)::A_{id}(x)$	$\mathbb{R}_{id}(\beta)::A_{id}(y)$
$\mathbb{R}_{id}(\alpha).precondition:= A_{id}(x).State(a)$	$\mathbb{R}_{id}(\beta).precondition:= A_{id}(y).Input$
$\mathbb{R}_{id}(\alpha).result:= A_{id}(x).State(b)$	$\mathbb{R}_{id}(\beta).result:= A_{id}(y).Output$
$<\mathbb{R}_{id}(\alpha),A_{id}(x),A_{id}(x).State(a),A_{id}(x).State(b)>$	$<\mathbb{R}_{id}(\beta),A_{id}(y),A_{id}(y).Input,A_{id}(y).Output>$

Annotation: '::' represents associated relationship, ':=' represents the assignment operator.

The rule mapping mechanism is shown as Figure 6. In addition, in order to make full use of the existing process fragments (otherwise known as composite services), each fragment can be mapped to one rule. Developers or users can manage the entire Rule Base by the component of Rule Management in the User Interface. The changes of business policy could be achieved by modifying the relevant rules. When the current Rule Base is unable to generate solution, we can solve it by adding rules and injecting related services [6].

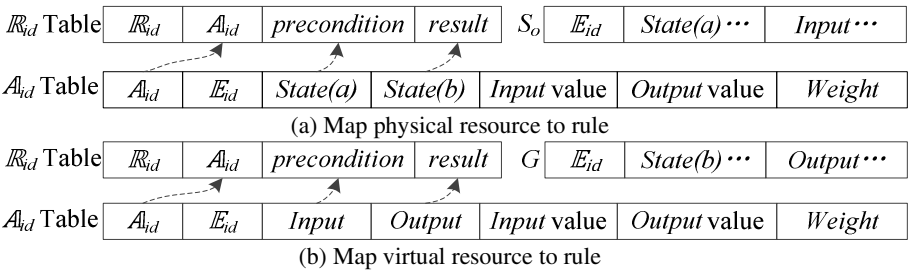

Fig. 6. The rule mapping mechanism

4.4 The RoDP Algorithm

The key problem of the RoDP is to generate the activity sequence to meet the business goals by matching the rules. The required activity sequence can be considered as the edge of the feasible path from current states to objective states in WDG. We propose two algorithms for generating activity sequence according to the number of the rules. One is a global approach that the Rule Engine creates WDG by matching all the rules and finds the optimal path from the current states to the objective states. The generated activity sequence is the optimal sequence in the optimal path. Another is a local approach that the Rule Engine matches the rule with the minimum weight from the current states to the objective states by greedy algorithm step by step. This approach is to solve the suboptimal path, when the number of the rules is very large, and the generated activity sequence is the suboptimal sequence.

The step to create the WDG is as follow: I. For the first rule, create the node S_{11} (S_{11}:=$\mathbb{R}_{id}(1)$.*precondition*) and the node S_{12} (S_{12}:=$\mathbb{R}_{id}(1)$.*result*) in WDG, then create the directed edge from S_{11} to S_{12}, and mark the edge with the weight $\mathbb{A}_{id}(x)$.*Weight* of related activity x; II. For the rule α (α>1), create the node $S_{\alpha 1}$ ($S_{\alpha 1}$:=$\mathbb{R}_{id}(\alpha)$. *precondition*) and the node $S_{\alpha 2}$ ($S_{\alpha 2}$:=$\mathbb{R}_{id}(\alpha)$.*result*) in WDG, merge the same nodes, then create the directed edge from $S_{\alpha 1}$ to $S_{\alpha 2}$, and mark the edge with activity weight $\mathbb{A}_{id}(y)$.*Weight*; III. Continue to II until α>m (m is the number of the rules).

1) Algorithm of optimal activity sequence (called RoDP-opt)

The key issue for generating the optimal sequence is to solve the optimal path from the current states to the objective states in the WDG. In our RoDP-opt algorithm, we use the Dijkstra algorithm to solve the optimal path.

The step to generate the optimal activity sequence is as follow: I. Extract the current states S_o (S_o::$\mathbb{E}_{id}(i)$.*State*) and the objective states G ($\mathbb{A}_{id}(x)$.*State(b)*, $\mathbb{A}_{id}(x+1)$.*State(b)*, …); II. Create the *G(V,E)* (namely WDG) according to all the rules; III. Label the nodes S_o and the nodes G in WDG; IV. Find the shortest path from S_o to S_n, and S_n is one of the elements of the set G; V. Output the optimal sequence based on the assessment strategy.

Algorithm 1. The RoDP-opt algorithm

```
capture the current context S_o and business goal G
receive user-preference
for rule R(α) (1≤α≤m)
    create V(G) and E(G), Edge[S_1][S_2]:= A_id.Weight
```

```
for Si∈V(G) (1≤i≤|V(G)|)
    if So= Si, then change the label Si to So
    else if Sn= Si, then change the label Si to Sn
 P`:= activity of (path of (Dijkstra(So,Sn)))
 P:=Min (P`)
 return P
```

2) Algorithm of suboptimal activity sequence (called RoDP-subopt)

It is very difficult to create and manage the WDG when the number of the rules is larger. The RoDP-subopt algorithm generates a suboptimal activity sequence by a local optimum manner. The key issue is to support backtracking in the solving process for generating the suboptimal sequence. In the RoDP-subopt algorithm, it is realized by emptying the related rules of the unreachable path temporarily and selecting the remaining rules which can be matched iteratively.

The step to generate the suboptimal activity sequence is as follow: I. For the current states S_o, match the *precondition* of all the rules with S_o, then select the activity x with minimum weight from all the matching rules, put activity x into activity sequence P and assign the *result* of the related rule to S_o; II. If there is not any matching rules with S_o, then backtracking (flag=0); III. Continue to I until $S_o \subseteq G$ or $P=\emptyset$.

Algorithm 2. The RoDP-subopt algorithm

```
capture the current context So and business goal G
receive user-preference
do
{   for rule R(α) (1≤α≤m)
      if Rid(α).precondition= So, then add Aid(x) in P`
  Aid(y):=Min(P`)
  add Aid(y) in P, P`:=∅, So:= Rid(β).result, flag:=0
  for rule R(α) (1≤α≤m)
      if Rid(α).precondition= So, then flag:=1
  if (flag=0)&(So⊈G), then
      delete Aid(y) in P, delete Rid(β), So= Rid(β).precondition
} while ((So⊈G)&(P≠∅))
recover Rid(β)
return P
```

4.5 The Process Assessment Strategy

Because of the different user preference, the generated activity sequence is different. This paper assesses the generated activity sequence with service cost, respond time, user experience or other QoS parameters. Users set the Process Assessment strategy by the Preference Selection in the User Interface, and the RoDP algorithm receives these parameters and generates corresponding activity sequence. The activity sequence can be directly executed by the Execution Engine, or it can be modified by developer in the Process Creation Environment.

$$\text{Optimal(P)} = \text{Min}\left\{\sum_{i=1}^{\theta}\left(\delta_i + \lambda_i + (\Delta - \tau_i)\right)\right\} \quad \text{s.t.} \begin{cases} \sum_{i=1}^{\theta} \delta_i \le C \\ \sum_{i=1}^{\theta} \tau_i \ge U \\ j \ne 3j \le n \\ \cdots\cdots, \cdots\cdots \end{cases}. \quad (1)$$

Formula 1 is the mathematical expression for calculating optimal sequence or sub-optimal sequence. δ, σ, and τ respectively represent the cost, response time, and user experience; θ is the number of activities in each group; Δ is the full mark of user experience ($0 \le \tau \le \Delta$); j (j:: A_{id}) is the serial number of the activity ($j \ne 3$ represents that no expect to use the activity which serial number is 3). Constrains (s.t.) provide more advanced assessment strategy for users. For example, if the preference is that users expect to choose activity sequence with minimum response time in the constraint that the total cost is not more than C, it can be expressed as formula 2.

$$\text{Optimal(P)} = \text{Min}\left\{\sum_{i=1}^{\theta} \sigma_i\right\} \quad \text{s.t.} \sum_{i=1}^{\theta} \delta_i \le C\,. \quad (2)$$

4.6 Experiment and Analysis

In this paper, we validate our RoDP approach by Matlab simulation experiment. We use service cost as the assessment strategy. To simplify the problem, we suppose that current state is S_o and objective state is S_n. The experimental data is shown as Figure 7, and the experimental result is shown as Figure 8.

R_{id}	A_{id}	*precondition*	*result*
0	29	S_o	S_1
1	14	S_o	S_3
2	9	S_o	S_2
3	21	S_1	S_4
4	36	S_1	S_6
5	1	S_2	S_5
6	11	S_3	S_5
7	13	S_4	S_7
8	6	S_4	S_n
9	23	S_5	S_6
10	5	S_5	S_8
11	31	S_6	S_n
12	17	S_8	S_n
…	…	…	…

A_{id}	E_{id}	*State(a)*	*State(b)*	*Weight*	activity
1	9	S_2	S_5	*2*	f
5	4	S_5	S_8	*5*	k
6	26	S_4	S_n	*5*	i
9	17	S_o	S_2	*3*	c
11	0	S_3	S_5	*1*	g
13	11	S_4	S_7	*2*	h
14	15	S_o	S_3	*2*	b
17	6	S_8	S_n	*3*	m
21	30	S_1	S_4	*4*	d
23	12	S_5	S_6	*2*	j
29	7	S_o	S_1	*1*	a
31	19	S_6	S_n	*3*	l
36	22	S_1	S_6	*5*	e
…	…	…	…	…	…

Fig. 7. The experimental data of RoDP

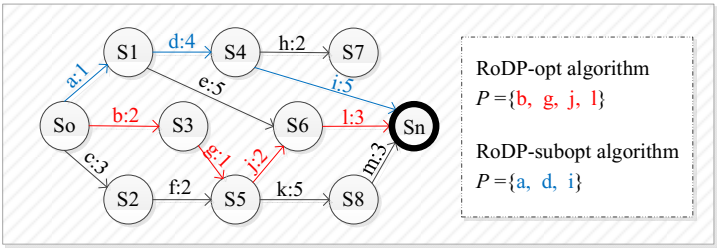

Fig. 8. The experimental result of RoDP

The computer configuration is that the CPU is dual-core 2.40GHZ and the memory is 4.00GB. We simulate the time consumption and space consumption while the rule number is 15, 50, 150, 450, 1350, and 4050, as shown in Figure 9. Abscissa indicates the number of rules. Experiments show that the time consumption is 7.162s for generating optimal activity sequence and the time consumption is 74.329ms for generating suboptimal activity sequence, when the number of rules is 1000. Through the experimental results, we conclude that we should use the RoDP-subopt algorithm to generate suboptimal activity sequence when the number of rules reaches 10^3 orders of magnitude. To avoid circular wait or infinite wait, we specify to execute the RoDP-subopt algorithm when execution time of the RoDP-opt algorithm exceeds 10s.

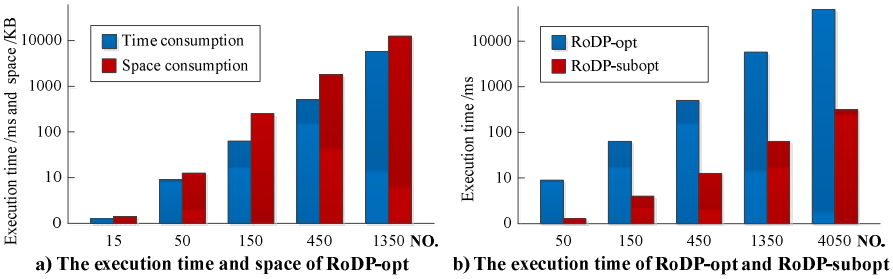

Fig. 9. The algorithm comparison under different number of rules

Table 1. The comparison of various methods

Method	Automatic Adjustment	Consistency Detection	Context Aware	New Requirement	User Preference	Manual Intervention
ADEPT		√			√	√
ConDec		√		√	√	√
Fragment-based	√		√			
Aspect-oriented			√		√	
ECA-based	√	√		√	√	
RoDP	√		√	√	√	√

Table 1 shows the comparison of various methods which were mentioned in the section of the Related Work. The major elements of comparison are as follows: whether to support the Automatic Adjustment at runtime; whether to support the Consistency Detection of the generated process; whether to respond to the changes of the Context and new requirement; whether to support the User Preference and Manual Intervention during the process of dynamic adaptation.

5 Conclusion

In the dynamic environment, it is an effective way to respond to the changes of environment information and application requirements by the dynamic planning approach. And the generated activity sequence of business process satisfies the business goals. This paper focuses on the dynamic assembly problem while we develop heating maintenance process based on the previously developed information service platform of heating industry. We propose the RoDP approach to adapt to the changes. The approach can realize adaptive adjustment of business process at runtime. The RoDP has high flexibility, because it is based on the rules. And we can also adjust the rules to respond to the changes. In addition, our approach supports user preference and manual intervention for the generated process. However, this paper does not focus on the data flow and the consistency detection which may lead to the problem of process inconsistency. So, we need to do further research on them. This approach not only solves the dynamic assembly problem between processes, but also proposes a general solution for the adaptation problem of business process in the changeable environment. The next step is to develop the corresponding components or tools.

Acknowledgments. This research is supported by the National Natural Science Foundation of China (Grant No. 61003067), National 973 Programs (Grant No. 2013CB329102, No.2012CB315802), and Key Project of National Natural Science Foundation of China (Grant No. 61132001), Program for New Century Excellent Talents in University (Grant No. NCET-11-0592), The technology development and experiment of innovative network architecture (CNGI-12-03-007).

References

1. Aalst, W.M.P., Weske, M., Grunbauer, D.: Case handling: a new paradigm for business process support. Data & Knowledge Engineering 53(2), 129–162 (2005)
2. Adams, M., ter Hofstede, A.H.M., van der Aalst, W.M.P., Edmond, D.: Dynamic, extensible and context-aware exception handling for workflows. In: Meersman, R., Tari, Z., et al. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 95–112. Springer, Heidelberg (2007)
3. Bucchiarone, A., Marconi, A., Pistore, M., et al.: Dynamic adaptation of fragment-based and context aware business processes. In: 2012 IEEE 19th International Conference on Web Services (ICWS), pp. 33–41. IEEE (2012)
4. Bucchiarone, A., Pistore, M., Raik, H., et al.: Adaptation of service-based business processes by context-aware replanning. In: 2011 IEEE International Conference on Service Oriented Computing and Applications (SOCA), pp. 1–8. IEEE (2011)

5. De, L.M.: Adaptive process management in highly dynamic and pervasive scenarios. arXiv preprint arXiv: 0906.4149 (2009)
6. Guinard, D., Trifa, M., Karnouskos, S., Spiess, P., Savio, D.: Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. IEEE Transactions on Services Computing 3, 223–235 (2010)
7. Heinl, P., Horn, S., Jablonski, S., et al.: A comprehensive approach to flexibility in workflow management systems. In: ACM SIGSOFT Software Engineering Notes, vol. 24(2), pp. 79–88. ACM (1999)
8. Kammer, P.J., Bolcer, G.A., Taylor, R.N., et al.: Techniques for supporting dynamic and adaptive workflow. Computer Supported Cooperative Work 9(3-4), 269–292 (2000)
9. Mejia Bernal, J.F., Falcarin, P., Morisio, M., et al.: Dynamic context-aware business process: a rule-based approach supported by pattern identification. In: Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 470–474. ACM (2010)
10. Papazoglou, M.P., Traverso, P., Dustdar, S., et al.: Service-oriented computing: State of the art and research challenges. Computer 40(11), 38–45 (2007)
11. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
12. Pfeffer, H., Linner, D., Steglich, S.: Dynamic adaptation of workflow based service compositions. In: Huang, D.-S., Wunsch II, D.C., Levine, D.S., Jo, K.-H. (eds.) ICIC 2008. LNCS, vol. 5226, pp. 763–774. Springer, Heidelberg (2008)
13. Reisig, W., Rozenberg, G. (eds.): APN 1998. LNCS, vol. 1491. Springer, Heidelberg (1998)
14. Ruy, S.H., Casati, F., et al.: Supporting the dynamic evolution of Web service protocols in service-oriented architectures. ACM Transactions on the Web 2(2), 1–45 (2008)
15. Schonenberg, H., Mans, R., Russell, N., et al.: Process flexibility: A survey of contemporary approaches. In: Dietz, J.L.G., Albani, A., Barjis, J. (eds.) Advances in Enterprise Engineering I. LNBIP, vol. 10, pp. 16–30. Springer, Heidelberg (2008)
16. Vaculin, R., Heath, T., Hull, R.: Data-centric Web Services Based on Business Artifacts. In: IEEE 19th International Conference on Web Services, pp. 42–49. IEEE (2012)
17. Verma, K., Gomadam, K., Sheth, A.P., et al.: The Meteor-S approach for configuring and executing dynamic web processes. Lsdis Meteors project. Technical report (2005)
18. Wanf, Y., Yang, J., Zhao, W.: Change impact analysis for service based business processes. In: 2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp. 1–8. IEEE (2010)
19. Weske, M.: Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences, pp. 1–10. IEEE (2001)
20. Yu, J., Han, J., Sheng, Q.Z., Gunarso, S.O.: PerCAS: An approach to enabling dynamic and personalized adaptation for context-aware services. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) Service Oriented Computing. LNCS, vol. 7636, pp. 173–190. Springer, Heidelberg (2012)