TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.1, January 2014, pp. 442 ~ 449
DOI: http://dx.doi.org/10.11591/telkomnika.v12i1.4148

Research on Adaptation of Service-based Business Processes

Zongmin Shang
Animation and Art Department, Zibo Vocational Institute, China
West of Liantong road, Zibo, Shandong, 255314, China Ph/Fax: +86533-2828208/2828209
e-mail: shangzongmin@mail.sdu.edu.cn

Abstract

This paper proposes an adaptation mechanism based on adaptation planning graph for service-based business processes. First, a three-layer representation model of service-based business process is introduced. Second, control-flow patterns of tasks, goal, logic model of service-based business process and adaptation planning graph are introduced to enforce reliability of composite web services at run-time. Finally, a simulation example of adaptation in service-based business processes is given. Simulations prove that this approach can efficiently guarantee the reliability of composite services at run-time.

Keywords: *service-based business process, adaptation planning graph, web service*

1. Introduction

In recent years, service-oriented architectures have been widely used for the realization of complex business processes. A business process consists of a group of business activities undertaken by one or more organizations in pursuit of some particular goal [1-2], and those activities are realized through the invocation of a set of available services. One of the key ideas underlying the service-oriented paradigm is that of allowing the combination of existing business activities, to obtain new services from different third-party organizations that satisfy some given requirements and goals.

Modern business processes often operate in dynamic, open and non-deterministic environments [3]. Dynamic context changes or undesirable outcome of some activities may often cause abnormal termination of the process and prevent the achievement of the business goals. Adaptation in process management systems is the key to their successful applicability in practice, and adaptation of the business process can be involved in dynamically at design time or at run time. But the run time modification of the business process is the main problem which has attracted serious concern of many researches [1-4]. This requires enriching processes with goals which specify what is pursued by the process execution.

This paper addresses the problem how to adapt a business process correctly while it involves different heterogeneous services, especially at running time. Our approach is based on a three-layer representation, where the first layer describes the specific business process with composition tasks, the second layer describes the abstract flow in terms of the goals that composition tasks should achieve, and the third layer is the concrete flow with composition web services to achieve the goals.

The rest of the paper is structured in the following way. In Section II, we present a motivating example and describe our three-layer representation, and describe the elements involved: Control-Flow Patterns of Tasks in a Service-based Business Process, Goal, Logic Model of Service-based Business Process, the definition of Adaptation Planning Graph is given in the end of section 2. A simulation experiment is described in section 3 to address adaption of service-based business processes. Finally, we discuss related work in section 4 and provide some concluding remarks in section 5.

2. Application Representation

In order to illustrate the need for a new approach to service composition we use a variant of a well-known travel domain scenario, which appears in various forms in the literature

Received June 28, 2013; Revised August 29, 2013; Accepted September 20, 2013

on service composition for demonstration purposes. In such scenario, we aim to provide a composed service that can deliver to the users and manage travel packages consisting of flight tickets and hotel reservations.

For modeling a service-based business process, we consider that there exist three layers (Figure 1), where the first layer describes the specific business process with composition tasks, the second layer describes the abstract flow in terms of the goals that composition tasks should achieve, and the third layer is the concrete flow with composition web services to achieve the goals.

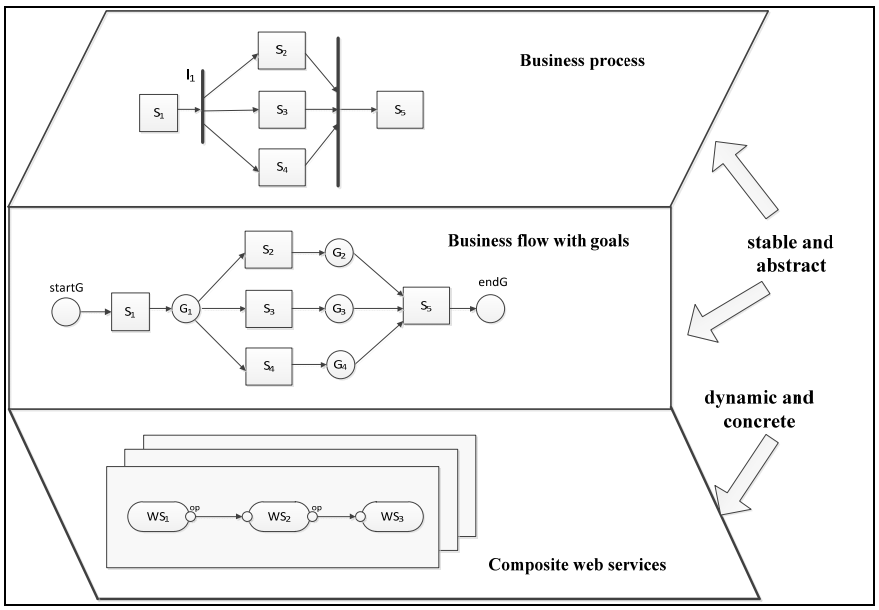

Figure 1. Application representation

In this application representation, the first layer is thus the domain knowledge, which is stable and abstract. The second layer is the stable part of the process knowledge, represented using goals, while the third is the dynamic and concrete part, represented using composition web services.

To understand the kind of problems we want to tackle and the approach to do so we present an explanatory scenario. A client wants to arrange a tour during her holiday; she would like to make a tour of city A and city B, book for a hotel room, rent a car, and buy either a train ticket or a plane ticket. Therefore, she will connect to a travel agency, where she specifies her request and in her turn she will receive a travel plan that realizes it. As shown in Figure 2, it shows that the tour is started with goal *start G*, service S_1 (by airplane, flight K), goal G_1, and then, service S_2 (sight-seeing route Y), goal G_2, service S_3 (accommodation in hotel B), goal G_3, service S_4 (rent car T), goal G_4, and finally service S_5 (by airplane, flight M), goal *endG*. Services S_2, S_3 and S_4 can be executed concurrently.

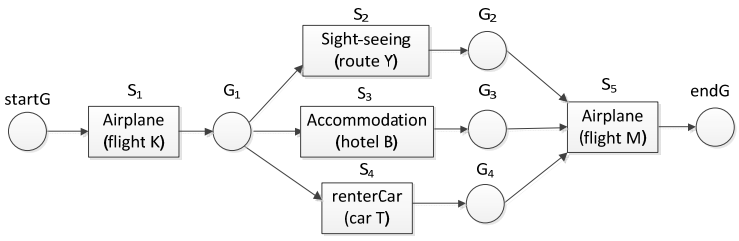

Figure 2. An example of service-based business process

2.1 Control-Flow Patterns of Tasks in a Service-based Business Process

We describe a Service-based Business Process with relations among tasks through control-flow patterns. As described in [5], a pattern “is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts”. The application of a patterns-based approach to the identification of generic workflow constructs was proposed in [5], which identified several control-flow patterns relevant to the control-flow perspective of workflow systems. For our purpose, patterns address business requirements in an imperative control-flow style expression, but are removed from specific languages. We employ seven control-flow patterns to establish a formal basis for understanding the requirements of a business process.

Pattern 1 (Sequence)

The web service to implement a task in a service-based business process is enabled after the completion of a web service that implements the preceding task.

Pattern 2 (Parallel)

A service-based business process diverges into two or more parallel branches, each of which execute concurrently.

Pattern 3 (Synchronization)

The convergence of two or more branchesinto a single subsequent branch.

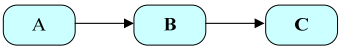

Figure 3. Sequence pattern

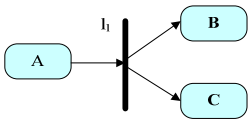

Figure 4. Parallel pattern

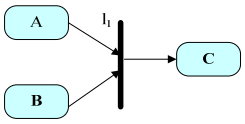

Figure 5. Synchronization pattern

Pattern 4 (Interleaved Parallel Routing)

A set of web services to implement corresponding tasks of two or more parallel branches can run in an interleaving way.

Pattern 5 (Interleaved Sequence Routing)

A set of web services to implement corresponding tasks has a free partial ordering defining the requirements with respect to the order in which they must be executed. Each web service in the set must be executed once and they can be completed in any order. However, as an additional requirement, no two web services can be executed at the same time.

Pattern 6 (Including Routing)

The web service to implement a task in a service-based business process should start after the start of a web service that implements the preceding task, and complete before the completion of a web service that implements the preceding task.

Pattern 7 (Intercrossing Routing)

The web service to implement a task in a service-based business process should start after the start of a web service that implements the preceding task, and complete after the completion of a web service that implements the preceding task.

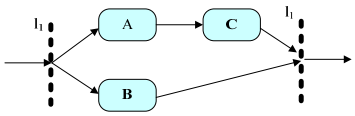

Figure 6. Interleaved parallel routing pattern

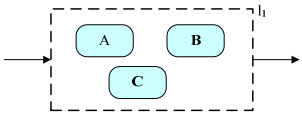

Figure 7. Interleaved sequence routing pattern

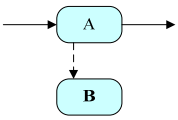

Figure 8. Including routing pattern

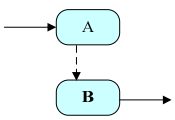

Figure 9. Intercrossing routing pattern

Definition 1 (States of Tasks in a service-based Business Process)

States of tasks in a service-based business process can be divided into {“initial”, “active”, “failed”, “aborted”, “cancelled”, “completed”} shown as Figure 10.

The initial sate of a task is “initial”, denoting the web service to implement this task is ready. While a web service has been executing, the state of the corresponding task will change to “active”. The sate of a task will change to “completed” if the correlative web service completed successfully, and “failed” if the execution failed. A web service can be cancelled if it had not started, the state of the corresponding task will change to “cancelled”. A web service can also be aborted if it is executing, and the state of the corresponding task will change to “aborted”.

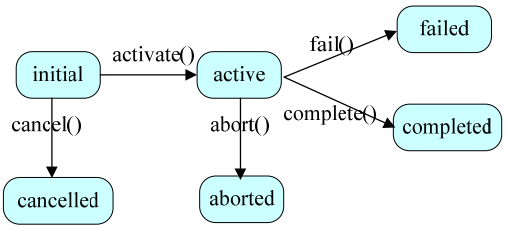

Figure 10. Transition Graph of Tasks in a service-based business process

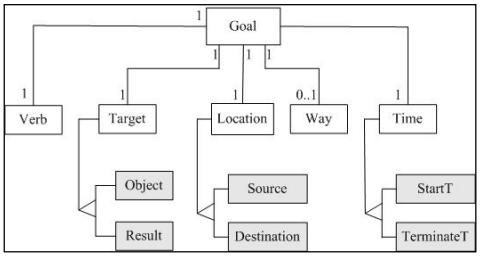

Figure 11. The Goal structure

2.2 Goal

In a service-based business process, a goal indicates the target to be achieved by the current service(s) and the conditions to be satisfied by the next service(s). The goal can be represented using either an imperative or a declarative language. Imperative process models focus on the way to achieve the goal, and assume that the environment is stable. Declarative process models specify the goal through constraints which approximate the desired behavior. Such descriptions are suitable for frequently changing environments, but cannot be executed completely automatically [6].

Definition 2(Goal) Goal in a service-based business process is defined as a tuple, like depicted in Figure 11, Goal := (*Verb, Target, Location, Way, Time*). In which,

The *Verb* describes the action to achieve a goal. *Target*=<*Object*, *Result*>. The *Target* designates entities affected by the goal. An *Object* is supposed to exist before the goal is achieved, and a *Result* will be satisfied after the goal has been achieved. *Result* can be of two kinds: 1) entities which do not exist before the goal is achieved and 2) abstract entities which exist but are made concrete as a result of goal achievement. *Location*=<*Source*, *Destination*>. The two types of *Location* identify respectively the initial and final location of services to be communicated. The *way* specifies the means to satisfy a goal. *Time*=<*StartT*, *TerminateT*>. The two types of *Time* identify respectively the start and end time of services to be executed.

Example 1 This example lists the description about a goal G_1 in a service-based business process.

```
<?xml version =”1.0” encoding=”UTF-8”>
<Goal Name= G1>
    <Verb>take</Verb>
    <Target>
        <Object>Flight K</Object>
        <Result>weather, health</Result>
    </Target>
    <Location>
        <Source> airport S, city A </Source>
        <Destination> airport m, city B </Destination>
    </Location>
    <Way>by plane </Way>
    <Time>
        <StartT> Sat,13:30 </ StartT>
        <Deadline> Sat,16:10 </Deadline>
    </Time>
</Goal>
```

The description defines the goal G_1 as: (Take) $_{Verb}$ (Flight K) $_{Object}$ from (airport S of city A) $_{Source}$ to (airport m of city B) $_{Destination}$ (by airplane) $_{Way}$, the plane will take off at (Sat, 13:30) $_{StartT}$ and land on at (Sat, 16:10) $_{TerminateT}$. The results of (weather and health) $_{Result}$ will be given dynamically during the flight.

2.3 Logic Model of Service-based Business Process

In the web service environment, a business process can be modeled using the BPEL4WS standards, where the whole business process can be regarded as a composite web service with each task corresponding to an operation of a component web service [7]. In this work we model a business process in the same way, where the whole business process is regarded as a set of tasks with each task corresponding to a goal, and each goal is achieved by an operation of composite web services.

Definition 3 (Logic Model of Service-based Business Process)

The can be modeled as an acyclic directed graph in the form of G(*T*, *L*, f_{in},f_{out}, *WP*, f_s,t_s), where

1) $T = \{t_1, t_2, ..., t_m\}$, $t_i \in T (1 \le i \le m)$ represents a task in the service-based business process. A task is implemented by a correlative web service.

2) $L = \{l_1, l_2, ..., l_m\}$, $l_i \in L (1 \le i \le m)$ represents the conjunction among tasks. $l_i.type \in$ {“synchronization”, “interleaved parallel routing”, “interleaved sequence routing”}.

3) $f_{in} : T \to Type$ is a mapping function that represents the conjunction between a task and its preceding tasks, where *Type*={“Null”, “Sequence”, “Parallel”, “Synchronization”, “Interleaved Parallel Routing”, “Interleaved Sequence Routing”, “Including Routing”, “Intercrossing Routing”}. For the starting task t_s, $f_{in}(t_s)$=“Null”.

4) $f_{out} : T \to Type$ is a mapping function that represents the conjunction between a task and its succeeding tasks, where *Type*={“Null”, “Sequence”, “Parallel”, “Synchronization”, “Interleaved Parallel Routing”, “Interleaved Sequence Routing”}. For the end task t_d, $f_{out}(t_d)$=“Null”.

5) $WP \subseteq T \times T \times \prec_{adapt}$ indicates the set of adaptation dependency of tasks. $< t_i, t_j, \prec_{adapt} > \in WP$ represents that if the service which implements task t_j needs to be adapted, the service which implements task t_i needs also to be adapted.

6) $f_s : T \to States$ is a function which maps each task in set *T* to a certain kind of states in set *States*, where *States*={“initial”, “active”, “failed”, “completed”, “aborted”, “cancelled”}. States of all tasks are “initial” before a service-based business process start running.

7) t_s indicates the starting task, which $prec(t_s) = \text{"null"}$.

In addition, we give several functions as follows:

- $prec, succ : T \to 2^T$ are functions which define for each task $t_i \in T$ its preceding tasks and succeeding tasks respectively. t_i is said to be the preceding task of t_j when it exists that $t_i \in prec(t_j)$. t_i is said to be the succeeding task of t_j when it exists that $t_i \in succ(t_j)$.
- $f_{IPR} : T \to 2^T$ is a function which gets the tasks whose conjunction with a task in set *T* is “Interleaved Parallel Routing”. $f_{IPR}(t_i) = null$ indicates that there is no task in set T whose conjunction with t_i is “Interleaved Parallel Routing”.
- $f_{ISR} : L \to 2^T$ is a function which gets the tasks whose conjunctions is “Interleaved Sequence Routing” by L.
- $f_{Includ} : T \to 2^T$ is a function which gets the tasks whose conjunction with a task in set *T* is “Including Routing”. $f_{Includ}(t_i) = null$ indicates that there is no task in set T whose conjunction with t_i is “Including Routing”
- $f_{Cross} : T \to 2^T$ is a function which gets the tasks whose conjunction with a task in set *T* is “Intercrossing Routing”. $f_{Cross}(t_i) = null$ indicates that there is no task in set T whose conjunction with t_i is “Intercrossing Routing”.

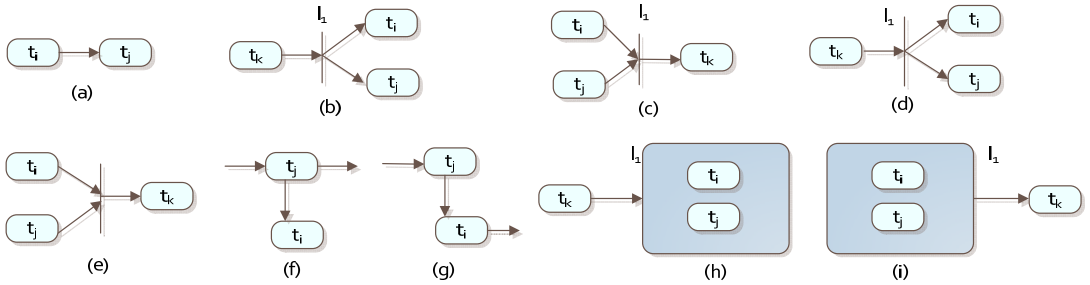

Figure 12. Dependency among tasks

Now we give seven formulas according to conjunctions among tasks in a service-based business process.

(F1) $t_j \prec_{adapt} t_i$ iff: $f_{out}(t_i) = \text{"sequence"} \wedge f_{in}(t_j) = \text{"sequence"} \wedge t_j \in succ(t_i) \wedge t_i \in prec(t_j)$. As shown in Figure 12(a).

(F2) $(t_i \wedge t_j) \prec_{adapt} t_k$ iff: $f_{out}(t_k) = \text{"parallel"} \wedge \{t_i, t_j\} \in succ(t_k)$. As shown in Figure 12(b).

(F3) $t_k \prec_{adapt} (t_i \vee t_j)$ iff: $f_{in}(t_k) = \text{"synchronization"} \wedge \{t_i, t_j\} \in prec(t_k)$. As shown in Figure 12(c).

(F4) $(t_i \wedge t_j) \prec_{adapt} t_k$ iff: $f_{in}(t_i) = f_{in}(t_j) = \text{"Interleaved Parallel Routing"} \wedge (t_i, t_j) \in succ(t_k)$. As shown in Figure 12(d).

(F5) $t_k \prec_{adapt} (t_i \vee t_j)$ iff: $f_{out}(t_i) = f_{out}(t_j) = \text{"Interleaved Parallel Routing"} \wedge (t_i, t_j) \in prec(t_k)$. As shown in Figure 12(e).

(F6) $t_i \prec_{adapt} t_j$ iff: $f_{in}(t_i) = \text{"Including Routing"} \wedge t_j \in prec(t_i) \wedge f_s(t_i) = \text{"initial"}$. As shown in Figure 12(f).

(F7) $t_i \prec_{adapt} t_j$ iff: $f_{in}(t_i) = \text{"Intercrossing Routing"} \wedge t_j \in prec(t_i) \wedge f_s(t_i) = \text{"initial"}$. As shown in Figure 12(g).

(F8) $(t_i \wedge t_j) \prec_{adapt} t_k$ iff: $f_{out}(t_k) = f_{in}(t_i) = f_{in}(t_j) = \text{"Interleaved Sequence Routing"} \wedge f_{ISR}(l_1) = \{t_i, t_j\} \wedge succ(t_k) \in f_{ISR}(l_1)$. As shown in Figure 12(h).

(F9) $t_k \prec_{adapt} (t_i \vee t_j)$ iff: $f_{in}(t_i) = f_{in}(t_j) = f_{in}(t_k) = \text{"Interleaved Sequence Routing"} \wedge f_{ISR}(l_1) = \{t_i, t_j\} \wedge prec(t_k) \in f_{ISR}(l_1)$. As shown in Figure 12(i).

Definition 4 (Adaptation Planning Graph, APG) Adaptation Planning Graph can be modeled as a directed graph in the form of APG= (AWS, AD, AC), where

1) AWS indicates a set of adaptation web services as shown in definition 2.

2) AD indicates a set of adaptation dependencies.

Dependency type includes compensation dependency, cancellation dependency, alternative dependency, and transfer dependency. For simple, if there exists $ws_i \rightarrow_{AD} ws_j$, then it also exists $aws_i \rightarrow_{AD} aws_j$.

3) AC is a set of adaptation costs for all adaptation web services in AWS . For the web services *ws* that implements an adaptation task, its cost can be calculated with formula:

$$AC(ws) = \sum_{i=1}^{n} \left[\frac{w_i}{\sum_{k=1}^{n} w_k} \cdot \frac{Q_i^A - Q_i}{Q_i^{\max}} \right]$$

In which, *n* indicates numbers of cost's properties, $w_i \in [0,1]$ indicates the weight of the i_{th} property. Q_i^A is the value of the i_{th} property that clients can be accepted, $Q_i^{\max}$ is the value of

the i_{th} property that is established in advance. The more value of $AC(ws)$, the more according with clients acceptance. It means going beyond acceptance when the value of $AC(ws)$ is negative.

3. A Simulation Experiment

We gave a simulation scene to evaluate the proposed approaches for adaptation during a service-based business process execution. Firstly, we give the simulation scene as following: With the help of Service-based Business Process Application Platform, Mr. Zhang got a travel plan to visit Beijing on the weekend. He will start from Shanghai Hongqiao Airport at 7:00 pm on Friday by flight MU5123 of China Eastern Airlines, arrive at Beijing Capital International Airport at 9:20 pm. He will then take an airport shuttle bus to Beijing Jingyuan Hotel at 10:30 pm. Mr. Zhang will visit the Palace Museum and Tian'anmen Square on Saturday morning, and visit the National Stadium and the National Aquatic Center on the afternoon. He will go shopping at Saturday night near Xidan Shopping Center. Mr. Zhang will check out at 6:00 am of Sunday, and take the car he has rented before to visit the Great Wall and the Ming Tombs. The car he takes will arrive at Beijing Capital International Airport at 5:00 pm. Mr. Zhang will take the Flight CA1549 of Air China at 6:30 pm, and arrive at Shanghai Hongqiao Airport at 8:40 pm on Sunday. Figure 13 shows the service-based business process andcorresponding methods of adaptation handling.

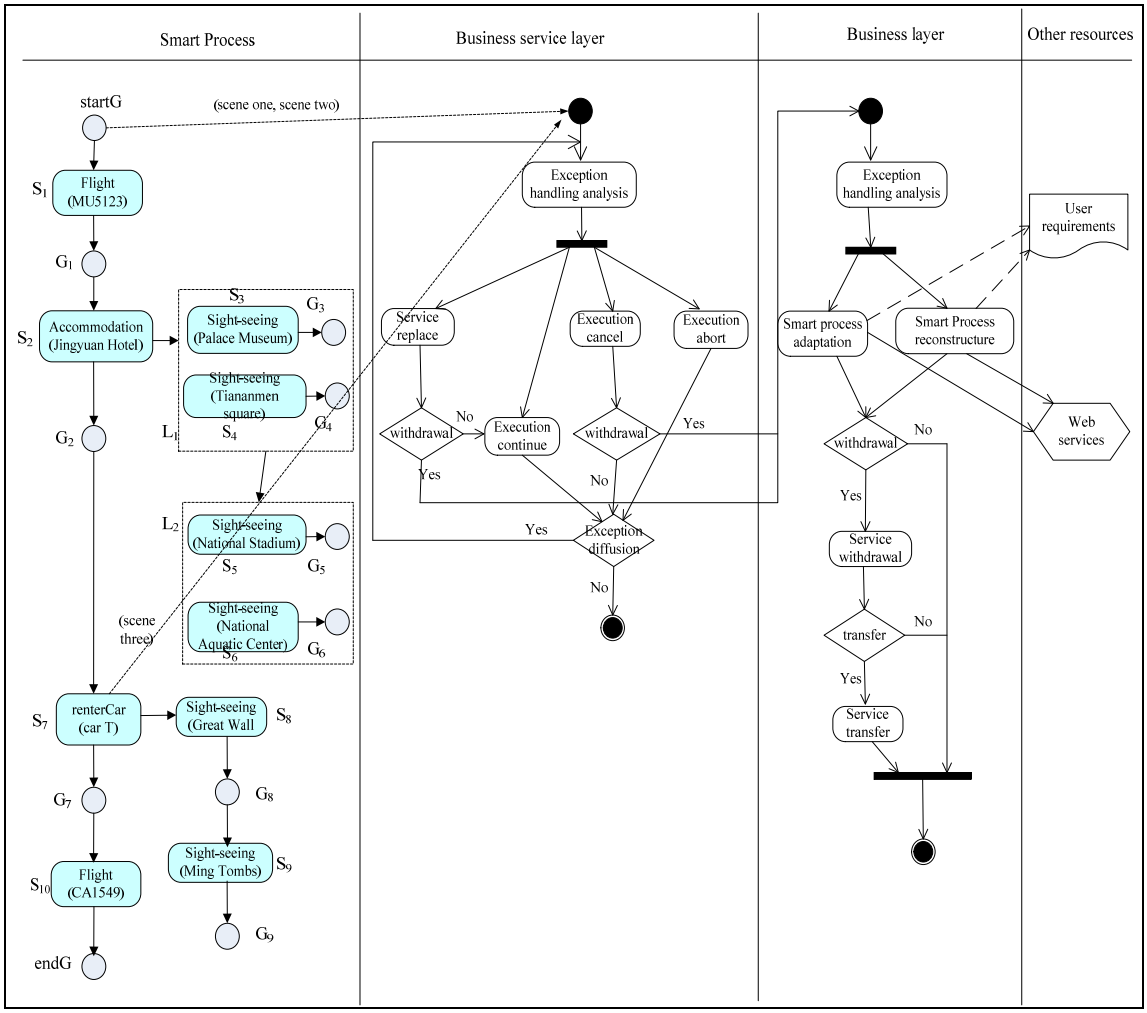

Figure 13. Adaptation handling mechanism

4. Related Work

Adaptation in process management systems is the key to their successful applicability in practice. The most trivial approach to achieve adaptation consists in analyzing at design time all the extraordinary situations, developing corresponding recovery activities, and embedding them into a reference process, using standard mechanisms such as exception handling or dedicated mechanisms for encoding adaptation into business process languages [8-9].

Much of the earlier work on adaptation concentrated on manually changing traditional processes at both the logic and instance level. Using the ADEPT framework [10] the process is adapted in an ad hoc manner according to the rule triggered. Based on ADEPT2, Dadam et al. [11] illustrates how ad-hoc changes of single process instances as well as process schema changes with (optional) propagation of these changes to the running instances are supported in an integrated, safe, and easy-to-use manner.

5 Conclusion and Future work

We presented an approach for composing pervasive process fragments according to context and goals logic model of service-based business process

In this paper, based on the logic model of service-based business process, we presented an approach to handle adaptation when composite web services are running. Furthermore, Adaptation Planning Graph (APG) is used to adapt some services when exception happened. In addition, compensation and transfer service mechanism is employed to deal with those services that cannot be compensated or retry times is too much and not to be accepted by clients.

In this paper, the concepts of transfer web service, compensation web service, and transaction handling idea are also based on our previous work [12-14].The present work makes certain assumptions that might not hold in some complex scenarios. In future, we will relax these assumptions to improve the applicability of our approach. In addition, the ability to deal with QoS-indicators is in our future work list.

References

[1] A Sirbu, A Marconi, et al. Dynamic Composition of Pervasive Process Fragments. *ICWS*. 2011: 73-80.
[2] Hermawan Hermawan. Riyanarto Sarno. Developing Distributed System With Service Resource Oriented Architecture. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(2): 389-399.
[3] B Antonio, P Marco, R Heorhi and K Raman. Adaptation of service-based business processes by context-aware replanning. *SOCA*. 2011: 1-8.
[4] P Bertoli, R Kazhamiakin, et al. Control Flow Requirements for Automated Service Composition. *ICWS*. 2009: 17-24.
[5] D Riehle and H Zullighoven. Understanding and Using Patterns in Software Development. *Theory and Practice of Object Systems*. 1996; 2: 3-13.
[6] N Russell, et al. Workflow Control-Flow Patterns: A Revised View. BPM Center Report (2006), BPMcenter.org.
[7] Z Yang and C Liu, Implementing a Flexible Compensation Mechanism for Business Processes in Web Service Environment. *ICWS*. 2006: 753-760.
[8] D Karastoyanova, et al. Extending BPEL for Run Time Adaptability. *IEEE International Enterprise Distributed Object Computing Conference*. 2005: 15-26.
[9] A Marconi, et al. Enabling Adaptation of Pervasive Flows: Built-in Contextual Adaptation. *Lecture Notes in Computer Science*. 2009; 5900: 445-454.
[10] R Manfred and D Peter. ADEPTflex-Supporting Dynamic Changes of Workflows without Losing Control. *Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems*. 1998; 10: 93-129.
[11] P Dadam, et al. Towards Truly Flexible and Adaptive Process-Aware Information Systems. *UNISCON*. 2008: 72-83.
[12] ZM Shang, Exception handling in Smart Process-based applications. *ICCASM*. 2010; v7522- v7526.
[13] ZM Shang. Research on exception handling of composite services based on compensation business process graph. *Jisuanji Xuebao/Chinese Journal of Computers*. 2008; 31: 1478-1490.
[14] LZ Cui, ZM Shang and YL Shi. A Transaction Management Model Based on Compensation Planning Graph for Web Services Composition. *ICWS*. 2011: 275-282.