PASCAL: A Protocol-based Approach for Service Composition and Dependable Optimization

Huipeng Guo
School of Economics and Management
Beihang University, Beijing, P. R. China
guohp@act.buaa.edu.cn

Xudong Liu, Zongxia Du, Jianxin Li, Mu Li
School of Computer Science and Engineering
Beihang University, Beijing, P. R. China
{liuxd, duzx, lijx, limu}@act.buaa.edu.cn

***Abstract*—Service composition is a promising technique for developing applications especially for those across multiple organizations. The dependability of composite services, however, is difficult to be guaranteed due to the distributed, dynamic and autonomous characteristics of service domains. In this paper, we address this problem by proposing a novel approach—PASCAL (Protocol-based Approach for Web Service Composition And DependabLe Optimization). We initially propose a protocol based service composition system architecture supporting online monitoring of services interaction and dynamic replacing of service. Then, we design the dependable optimization mechanism based on the service and dependencies monitoring. Furthermore, we design an algorithm to adjust the configuration adaptively base on policies and protocols to keep the composite service dependable while service failure appears. Finally we implement the PASCAL approach based on XService suit and evaluate the proposed mechanism through comprehensive experiments and achieve improved results.**

Keywords- service composition; protocol composition; service dependency; optimization algorithm; failure diagnosis

I. INTRODUCTION

Service orientied technology has been widely employed in a wide spectrum of applications. And service composition which constructs a composite service by composing basic Web services that may be distributed over the world has become the fundamental approach for web based application development. However, the dependability of services is not credible enough to support business applications especially to accomplish critical applications e.g., a disaster management system. When some component services fail or can not adapt to the changed environments, the users may severely suffer from the decreased quality.

The intrinsic dynamics of composite services stems from the fact that many component services usually belong to different providers distributed in the open Internet. Each component service is subject to different environments and changes, such as varying system load and available bandwidth. The properties of such a service may therefore change from time to time. In addition, the cooperate relationship among services or partners may change simultaneity. All these factors lead to the inevitable dynamic changes of the composite service. On the other hand, the new requirements posed by the users or the applications may evolve to meet different needs in reality. Furthermore, the general service composition modeling languages use structures, such as sequence or parallel to descript synchronization constraints among cooperative services, and the structures are imperative and often obfuscate the sources of dependencies [1]. Besides nested structures and scattered code will be produced, especially when concurrency exists [1]. As a result, it will be difficult to add or delete a constraint in a process, and it will be difficult for composite service to update, modify and reuse.

To adapt to the efficient development and evolution requirement in the open, dynamic environment, the variety of services and relationship among services must be concerned and adaptive method should be used. The service composition has been studied and several attempts have been made to construct adaptive composite services [2-5]. However, it is still difficult to support the dependable composition and optimization of services dynamically.

In this paper, we propose novel interaction centric system named PASCAL to optimize service composition dynamically. Service dependencies are systematic categorized as data, protocol and policy for adaptive service composition and optimization. Based on dependency monitor, we propose a protocol based method supporting the failure diagnosis and adjustment dynamically. To ensure the dependability, through the combination of evolution policy and interaction protocol, we design algorithms to achieve the goals of failure recovery and dynamic adjustment.

We have made the following original contributions. First, we make the attempt to explore the approach of protocol based adaptive services composition. An innovative system PASCAL is proposed which implements adaptive service composition and optimization mechanism. Then, through the combination of evolution policy and interaction protocol, we design monitoring based failure diagnose algorithm for service recovery and adjustment. At last, we preliminarily implement PASCAL based on XService suite[4] and conducted comprehensive trace-driven experiments.

The rest of the paper is structured as follows. Section 2 describes the protocol based service composition method PASCAL. The protocol based optimization mechanism and algorithm is elaborated in Section 3. In Section 4, we discuss performance evaluation and analyze performance results. Finally, we conclude the whole paper in Section 5.

II. DESIGN OF PASCAL

In this section, we first introduce the relevant concepts and optimization model of composite services. Then, we describe the architecture of PASCAL.

A. PASCAL Model

The dependency and interaction relationships are critical to improve efficiency of services selection, to compose the services automatically as well as to diagnose failure. Due to complexity of interaction, some hidden dependencies cannot be captured by data and control dependency [1]. Base on the study of [1,6,7], considering of the practicality, we divide dependency into protocol dependency, policy dependency and data dependency. And then we establish a service composition and evolution model based on the dependency

Definition.1. (Dependency of service) Dependency of services *DoS* is defined as a tuple (*DataD*, *ProtocolD*, *PolicyD*), where *DataD* is the set of data dependencies of services, *ProtocolD* is the set of protocols, and *PolicyD* is the set of policy dependencies.

Data dependencies denote the definition-use relation among data producer and consumer in different messages [1]. For Example, the *amount* in message *AuthConfirm*(*Token*, *amount*) of the bank payment service has the same value with *fcost* in message *!fRe*(*fInfo*, *fcost*) of the flight reservation service. Protocol dependencies among services describe the set of protocols supported by services, which characterize the allowed message interaction sequences. Policy dependencies refer to a set of policies supported by services, and usually they are achieved on a set of rules.

Definition.2. (Protocol) A protocol *p* is defined as a triple (*Mes*, *Role*, *STE*), where *Mes* is the set of messages, defines the words and messages format used in protocol *p* ; *Role* is the set of participants in *p*; $STE = (Q, Mes, \delta, q_0, F)$ is the interactive state machine which describes the constraints of the message interaction between the participants in *p*, where *Q* is the set of the states; q_0 is the initial state and F is the set of final states; $\delta : Q \times Mes \rightarrow Q$ is the state transition function of the protocol *p*, in which $\delta(q, mes)=q'$, $m=(r, r', content(mes))$, $r, r' \in R$ means that the protocol changes from state *q* to state *q'* after the participant role r sends a message *m* to *r'*,. From the definition of protocol *p*, we can get the interaction behavior of all the roles in *p*.

Definition.3. (Role) In a protocol *p* = (*Mes*, *Role*, *STE*), each role *r* is defined as a tuple (Mes_r, STE_r), where Mes_r is the subset of messages sent or received by *r*, i.e., $Mes_r = \{mes \in Mes|\ mes=(r, r', content(mes))$ or $mes=(r', r, content(mes))\}$ and STE_r *i*s the subset of state transitions of *STE* triggured by messages in Mes_r.

Besides the standardization of message set and interactive state, protocols can also restrict behaviors of roles in the following perspectives. First, environment sssumption describes the environment related situations. For example an online purchasing protocol can assume that the acts such as refund and complaining are took charge by other protocols, so as to simplify the design of this protocol. Second, constraint describes a number of restrictive conditions of the interactive activities. For example, once there is no response to some request message in a specified interval, the execution of a protocol will be ended automatically.

We denote by P_s the set of all the protocols in which service *s* acts as a role. If a service must support multiple protocols at the same time, we have to design the appropriate composition of protocols. And the optimization of composite services includes the usage of different protocols and the axiom of protocol composition. So that composite services can adjust according to the actual demands in order to meet the different business goals.

Definition.4. (Component Service) Component service is defined as a triple $s = (OP_s, Role_s, CA_s, Q_s)$, where OP_s is a set of operations of the service *s*, $Role_s$ is the set of all roles of *s* such that each *p.r* in $Role_s$ means that service *s* serves as the role *r* in *p*. CA_s is a set of composition rules of service *s*, $Q_s=\{Q_s(p.r,\ p.r',\ ca)|\ p.r \in PR_s,\ p.r' \in R_p,\ Intera\ (p.r, p.r')=1,\ ca \in CA_s\}$ is the description of the non-functional attributes of service *s*, in which $Q_s(p.r,\ p\ r', ca)$ represents a tuple of QoS attributes values when service *s* serving as the role *p.r* interacts with the service serving as *p.r'*.

Definition.5. (Composite Service) Composite service *CS* is defined as a tuple (*S*, *DoS*(*DataD*, *ProtocolD*, *PolicyD*))where *S* is the set of component services of *CS*, *DoS* refers to the set of dependencies among services in *S*.

In this paper, the protocol emphasizes the nature of interaction and simplifies the implementation detail. And the details will be described based on the business policies of the partners. And for the composition of service, policies are defined by some composition constraint rule such as:

Composition Rule 1. For each protocol *p* in set *P*, there is a primary service in *S* for each role *r* in *p*.

Composition Rule 2. For every pair of roles r_1 and r_2 of protocol *p*, if r_1 and r_2 have the dependency *R*, their primary services s_1 and s_2 also meet the dependency *R*.

Besides, the users defined composition constraint rules reflects the function, behavior and quality requirement that need to be met by compose particular component services.

B. PASCAL Framework

PASCAL method relies on the interaction and dependency relationships among services. The structure of PASCAL is shown in Figure 1.

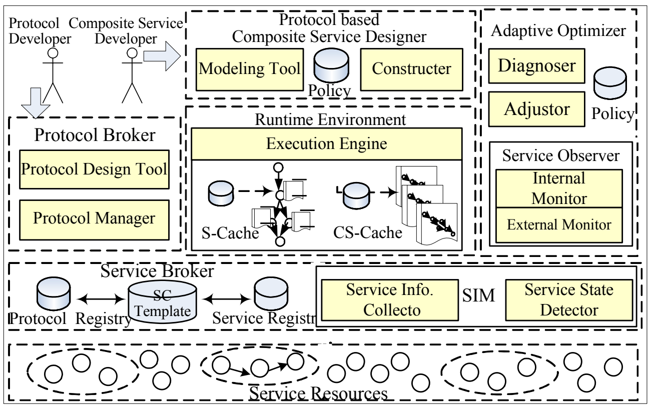


Figure 1. The PASCAL Framework

In PASCAL, the *Composite Service Designer* assists developers to define the abstract process, interactive relationship and interface between partners with the help of graphical tools. Then, in accordance with component protocols and protocol computation method [5], as well as the composition policies, services are assembled.

The *Protocol Broker* helps to design and manage the business protocols. The protocols determine how to interact among cooperation services, supports the autonomy of services and the realization flexibility, especially the flexibility to deal with exceptions or unexpected accidents.

In the *Service Broker*, the *Protocol Registry* stores public protocols. While the *Service Composition* (*SC) Template Registry* stores the composite service specification based on protocol. The *Service Registry* stores the service register information. The *Service Information Manager* (*SIM*) supports the evaluation of client monitoring, supports the proactive monitor to available states of services, detecting the quality of services under different policies.

In the *Runtime Environment*, the *Execution Engine* parses composite service documents built by *Constructer* and invokes the component services. The *Service Cache* [1, 8] is composed of component service cache named S-Cache and composite service cache named CS-Cache. They store backup services and support the switching of them.

Most important, the *Adaptive Optimizer* includes *Service Observer*, *Diagnoser* and *Adjustor*. The *Observer* monitors dependencies among services. Thereinto, the *Internal Monitor* observes the execution and interaction of composite services in use, records the quality and state data. While the *External Monitor* gathers the historical data of interaction. According to the interaction monitoring results and protocol specification, the *Diagnoser* can calculate the probability of failure causing by messages. Then the *Adjustor* figures out the optional policy via learning the varieties of the environment, completes service reconstruction and selection according to optimization policies. And the implementation of policies is based on our earlier work [8].

III. DEPENDABLE OPTIMIZATION MECHANISM

The PASCAL supports online failure diagnosis and adjustment so as to find the source message which causes failure and to optimize the composite service.

During the composition process, component services are seen as black-boxes, the source code and inner structure are invisible. While the protocol description can be used by the monitor, and it is designed in many business domains just as the PIPs in RosettaNet, which is described by UML and stored in the *Protocol Registry*. According to the protocols specification, the *Service Observer* can access interaction messages among services, check these messages, match the states with the state models and record or change them.

The process of diagnosis and recovery is shown in Fig.2 .When a service failure occurs, the monitor will check the *Protocol Registry* and composition policies, follow the state conversion rules to find out the services which have interaction with the service node. Then, a failure tree will be constructed by the failure tree generation algorithm based on [9]. The probability of every message that may cause a failure will be computed, corresponding service will be tested online or offline, the test results will be stored in a database, the QoS information will be updated and so on.

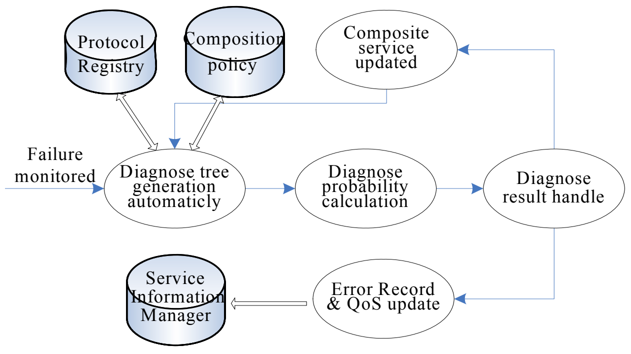

Protocol Registry
Composition policy
Composite service updated
Failure monitored
Diagnose tree generation automaticly
Diagnose probability calculation
Diagnose result handle
Service Information Manager
Error Record & QoS update

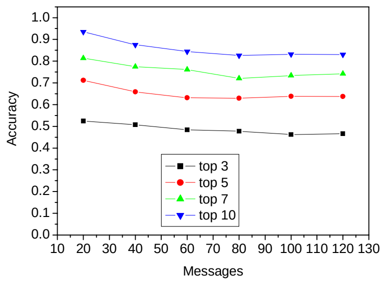
Figure2. Process of Failure Handling

To deal with the failure of component services in the runtime, we designed Service Failure Diagnosis Algorithm.

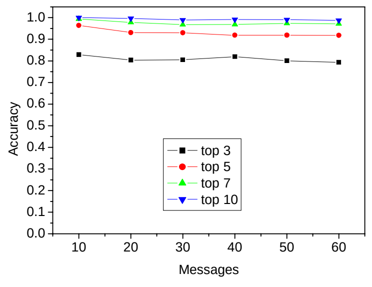
The basic idea of this algorithm is described as follows. First, define the data dependencies among services and input the messages interaction specification among operations (lines 3-6). Then, monitor and failure recovery policies of service composition will be loaded by Monitor (line 7). The dependencies will be monitored and the failure probability of operation and messages will be computed (line 8). When a failure occurs, failure probabilities of other messages (the compute method can be found in [9]) can be computed, the operations and messages will be analyzed in probability order (lines 11-14), then composite services will be adjusted according to restriction such as function, protocol and quality, and failure handle actions such as rollback or compensation will run (lines 16-19). At last, new composite service will be updated and be executed (lines 20-22).

Service Failure Diagnosis Algorithm

```
Input: initial composite service, data dependencies, protocol specification, service monitoring policies.
Output: updated composite service, dependencies
1. Initial(); //Initialization;
2. $\boldsymbol{cs} \leftarrow \boldsymbol{cs_0}$; //choose composite service to be monitored;
3. for all ($s_i$,$o_j$)
4.   $D_d$ ($s_i$, $o_j$, $data_k$)←($s_m$, $o_n$, $data_k$);
5.   $D_p$ ($s_i$, $o_j$, $message_k$)←($s_m$, $o_n$, $message_k$);
6. end for //set the data and protocol dependencies;
7. Read $MS_i$, $FR_i$ ();
8. Monitor and Calculate $P_k(s_i, o_j)$and $P_k(s_i, p_j, s_k)$;
9. Repeat
10.  While failures monitored;
11.   Query IOP &IOS, some log on for new; //Query failure history and probability of protocols and services;
12.   Generate the failure diagnosis tree with the failure service being the root;
13.  Calculate the Probability of correlative operation base on data and protocol dependencies;
14.  Sort the operations by the probability of failure;
15. for all corresponding($s_i$, $o_j$), do
16.   Test the message and operations;
17.   Search new services replace the service cause failure;
18.   Select(cs, fc, pc, qc); //following function, protocol and QoS constraint, select candidate services through HAF[3, 8] or other service selecting algorithm;
```



(a)



(b)

Figure 3. Diagnosis Accuracy vs. messages

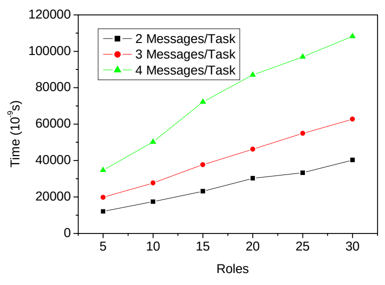


Figure 4. Diagnosis time vs. roles

```
19. Handle the operation of failure service;
20. Construct new business process cs';
21. cs ← cs';
22. Execute cs;
23. until cs is terminated
```

IV. Evaluation

We have implemented PASCAL system and evaluate the system through comprehensive experiments.

A. Environment and Metrics

PASCAL system is implemented based on our XServices suit [4] which provides a set of tools to enable SOA. And to evaluate the system, we adopt the metrics as follows.

Accuracy is defined as the ratio of the times that messages causing a failure are correctly found.

Diagnose Time is defined as the execution time of service messages diagnosing algorithm from the beginning till the probability list of messages cause failure is generated.

B. Evaluation

To evaluate our methods, we generate a certain number of services and interaction messages through the PASCAL.

The first experiment aims to measure the Accuracy of adaptive diagnose mechanism. 30 composite services are generated and the accuracy of execution for 1000 rounds is recorded. The results in Fig.3 (a) show that accuracy of composite services failure algorithm when average messages number is four in the protocols among services. With the number of top suspicious messages increase, the accuracy of diagnose become better. Even the front 3 messages are tested, accuracy is about 50%. And from Fig.3 (a) and (b), we can find the faulty messages more accurate if the interaction protocols among services are simpler.

The second experiment aims to evaluate the latency of the diagnose algorithm. The algorithm is deployed on a server with Intel Core Duo CPU 2.99GHz, 3.25GB RAM, Windows XP and Java 2 Edition V1.4.0. The average execution times of 1000 rounds for different number of messages are shown in Fig.4. We find that the diagnosis consumes more time, but it is still acceptable and the time increases very slowly as the number of messages increases.

Through our experiments, we can see that the dependency and interactive relationship among services are helpful to improve the efficiency and effectiveness of service searching, match and composition.

V. Conclusion

How to guarantee the dependability of composite service has become an important research issue. In this paper, we have designed an interaction-centric service composition system, PASCAL, supporting the online monitor of interaction and reconstruction of composite service adaptively. Then, we present diagnose mechanisms and corresponding algorithm to dynamic adjust the configuration base on policies and protocols. Finally we implemented the PASCAL method based on the XServices suit and evaluated the proposed mechanism.

Acknowledgments

This research was supported in part by China NSF grants (No.60903149), China 863 High-tech Programme (No.2009AA01Z419 and 2007AA010301) and China 973 Fundamental R&D Program (No.2005CB321803).

References

[1] Q. Wu, C. Pu, A. Sahai and R. S. Barga, Categorization and Optimization of Synchronization Dependencies in Business Processes, Proc. ICDE, Istanbul,Turkey, 2007, pp. 306-315.
[2] J. Harney and P. Doshi, Speeding up Adaptation of Web Service Compositions Using Expiration Times, Proc. WWW, Banff, Canada, 2007, pp. 1023-1032.
[3] H. Guo, J. Huai, Y. Li and T. Deng, KAF: Kalman Filter Based Adaptive Maintenance for Dependability of Composite Services, Proc. CAiSE, Montpellier, France, 2008, pp. 328-342.
[4] Xservices website, *http://forge.objectweb.org/projects/xservice*
[5] Y. Li, et al., QoS-aware Service Composition in Service Overlay Networks, Proc. ICWS, Utah, USA, 2007, pp. 703-710.
[6] G. Holzmann, Design and Validation of Computer Protocols [M]. Prentice Hall, 1991
[7] N. Desai, A. U. Mallya, A. K. Chopra, M. P. Singh, Interaction Protocols as Design Abstractions for Business Processes,IEEE Trans. on Software Engineering, vol. 31, Dec. 2005, pp. 1015-1027.
[8] H. Guo, et al., ANGEL: Optimal Configuration for High Available Service Compositio.Proc. ICWS, USA,2007, pp.280-287.
[9] G. Khanna, I. Laguna, F. A. Arshad and S. Bagchi, Distributed Diagnosis of Failures in a Three Tier E-Commerce System. Proc. SRDS, Beijing, China, 2007, pp.185-198.