The Journal of Systems and Software 86 (2013) 1089–1106

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Petri net based techniques for constructing reliable service composition

Guisheng Fan[a,b,*], Huiqun Yu[a,b], Liqiong Chen[c], Dongmei Liu[a]

[a] Department of Computer Science and Engineering, East China University of Science and Technology, China
[b] Shanghai Key Laboratory of Computer Software Evaluating and Testing, China
[c] Department of Computer, Science and Information Engineering, Shanghai Institute of Technology, China

ARTICLE INFO

Article history:
Received 3 September 2011
Received in revised form 6 June 2012
Accepted 18 November 2012
Available online 17 December 2012

Keywords:
Service composition
Petri net
Exception
Transaction attributes
Reliability

ABSTRACT

Service composition is an important mean for integrating the individual Web services to create new value added systems that satisfy complex requirements. However, it is challenging to guarantee the reliability of service composition in a distributed, dynamic and complex environment. This paper proposes an approach to constructing the reliable service composition. The underlying formalism is Petri net, which provides means to observe behaviors of basic component, and to describe their interrelationship. The transaction attributes, reliability and failure processing mechanisms are articulated. The composition mechanism systematically integrates these schemas into a transaction mapping model. Based on this, a reliable composition strategy and its enforcement algorithm are proposed, which can verify the behaviors of service composition at design time or after runtime to repair design errors. The operational semantics and related theories of Petri nets help prove the effectiveness of the proposed method. Finally, we use a simplified Export Service system to demonstrate the feasiability of the method.

1. Introduction

Service oriented Computing (SOC) is an approach to distributed computing that views software resources as dynamically discoverable services available on the Internet (Papazoglou and Heuvel, 2007). Because the function of single Web service is simple and limited, which cannot cater for the actual requirements, it is necessary to compose it to provide a more powerful service. Service composition provides a mechanism for distributed software integration, where different enterprise solutions cooperate to achieve a common goal, and primarily concerns the requests that cannot be satisfied by any available service, whereas a composite service obtained by combining a set of available Web services might be used (Shina et al., 2009). Thus, in recent years, some new Web Service frameworks (Francisco et al., 2009; Lee et al., 2009; Canfora et al., 2008) have been proposed for describing service composition.

With the increase of task service numbers, applications can be very complex in structure. These complex services such as Export Service system result from the aggregation of Web services offered by different organizations. As service composition is composed by different Web services invocation, where the results of Web service are passed to the next in an atomic manner. If a Web service fails the entire execution must be aborted, which will result in a waste of computing resources and slow response time. Furthermore, Web services are deployed on the unreliable Internet, and as they are often long running, loosely coupled, and cross administrative boundaries, it will require support for recovery and compensation, because the processes may be canceled, or services may be moved or withdrawn. Furthermore composite services are dynamic in that their components can be automatically selected at run-time based on specific requests, which makes the problem of reliability more challenging than conventional client–server systems. While many network applications such as financial services, online transaction or e-commerce are running in an unpredictable environment, and they requires a higher level of reliability. If the requests cannot be met, they will cause the loss of customers, economic loss, etc. Therefore, how to dynamically construct a reliable service composition according to the actual requirement has become a challenging problem.

Several researches have been done to address the requirements of transactional support for Web services. With centralized coordination as employed in Web services orchestration via WS-BPEL, a centralized process coordinator manages both the process flow and its transaction scopes (Jordan and Evdemon, 2007). To cater for the new features of transactions executed by Web services, various Web transaction specifications have been developed. Among the published specifications, WS-Coordination (Feingold and Jeyaraman, 2007), WS-AtomicTransaction (Little and Wilkinson, 2007) and WS-BusinessActivity (Freund and Little, 2007) are the most prominent ones. WS-Coordination specification describes an extensive framework for providing various coordination protocols. WS-AtomicTransaction is developed for simple

* Corresponding author at: Department of Computer Science and Engineering, East China University of Science and Technology, China. Tel.: +86 21 64253546.
E-mail addresses: gsfan@ecust.edu.cn (G. Fan), yhq@ecust.edu.cn (H. Yu), lqchen@sit.edu.cn (L. Chen), dmliu@ecust.edu.cn (D. Liu).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.11.037

and short-lived Web transactions, while WS-BusinessActivity for complex and long-lived business activities. These specifications do handle some basic issues of transactions, such as compensation, fault and exception handling. However, these specifications are inadequate to support the user-defined relaxed atomicity constraint in order to achieve a common goal, and they are unable to ensure reliability especially according to the users' specific requirements, such as failure atomicity and reliability. Accepted termination states (*ATS*) is a mean to express the required failure atomicity (Rusinkiewicz and Sheth, 1995).

It is important to consider these dynamic requirements in developing reliable service composition. In this paper, we investigate how to dynamically construct the reliable service composition based on the requirements of service consumers, and Petri nets are used as an underlying formalism. The main contributions of this paper are as follows.

First, we provide a flexible way for designers to specify their requirements in terms of control flow and correction (by specifying the set of accepted termination states (*ATS*) and reliability), then propose a service composition reliable net (*SCRN*), which is used to construct different components of service composition. We only need modify the specific mapping when the requirement changes, thereby increasing the reusability of the model. Meanwhile, we consider all the transactional combinations satisfying the global transactional requirements, the reliability of service is also characterized.

Secondly, we analyze the relationship between the *ATS* and the possibly reachable states of service composition. Then propose a reliable composition strategy and its enforcement algorithm, which is used to dynamically adjust available service to meet users' requirement under the premise of meeting the required transaction attributes. The redundancy technique is used to construct the invoked service set for each task, thus ensuring the reliability of service composition throughout the composite service life cycle.

Finally, the operational semantics and related theories of Petri nets help prove the effectiveness of proposed method at design time, and to analyze and check composite service execution reliability at runtime according to users' requirement. The results shown that the proposed method can not only meet the requirements of a service composition, it also inline with the current research trend.

The rest of this paper is organized as follows. In Section 2, we propose a system architecture and introduce the definitions underpinning our approach. We give the requirements of reliable service composition, and present a motivating example in Section 3 for the purpose of illustrating our results through the paper. In Section 4, we model for the different components of service composition, then propose a reliable composition strategy. The basic properties of constructed model are also analyzed. In Section 5, we prove the effectiveness of proposed method, the corresponding enforcement algorithm is also proposed. In Section 6, we show the implementation process and simulation results. Finally, Section 7 presents some related works while Section 8 is conclusion.

2. The framework of SCRN

Fig. 1 shows the system architecture for constructing reliable service composition. Its main components are explained as follows.

Part 1: Service layer, which is used to store all the semantic description of the registered Web services.

Part 2: Modeling and analysis, we use Petri nets to observe the behavior of service composition, and use the related techniques of Petri nets for analysis and verification, which includes two steps:

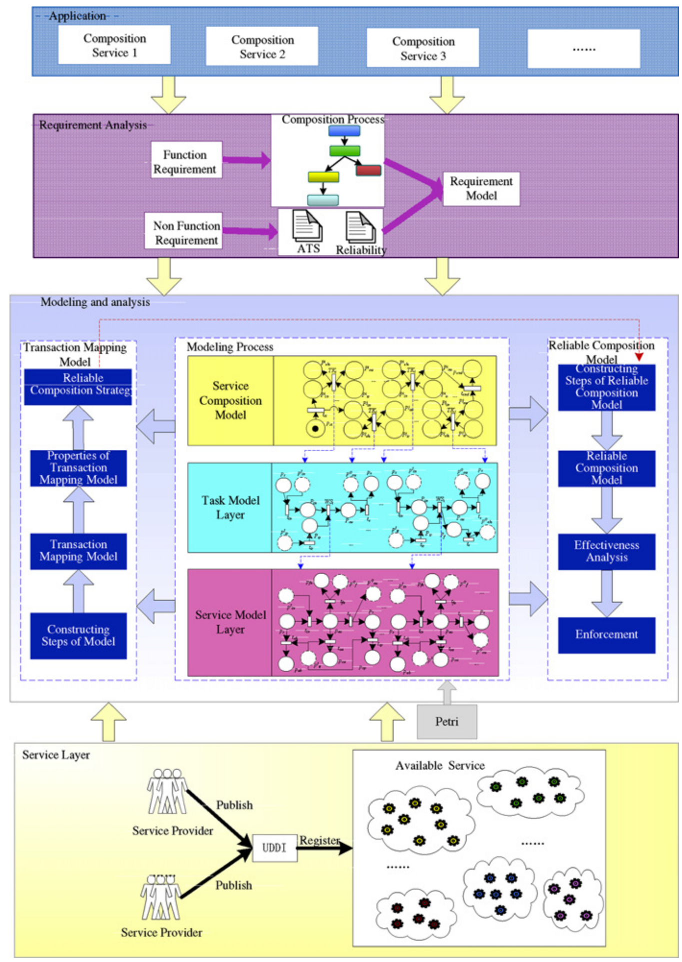

Fig. 1. System architecture.

Step 1: Modeling process, the different components of service composition are modeled by Petri net, which includes service, task, the relationship between tasks, service composition. The transaction attributes and reliability of service are also characterized. Then, a strategy for constructing reliable service composition is proposed based on the transaction attributes of service and users' requirements, thus getting the transaction mapping model of service composition.

Step 2: Analysis process, Petri nets and related theories are used to analyze the basic properties of constructed model. A reliable composition strategy and its enforcement algorithm are proposed, its effectiveness is also verified. Therefore, we can detect errors and optimize design in the early stages of software life cycle.

Part 3: Requirement analysis, the system will analyze the users' requirements, which includes the functional and non-functional requirement. In addition, it will assign tasks to computing nodes using the workflow management capability, thus forming the requirement model.

Part 4: Application layer, it provides an environment for service providers and consumers to represent their services and requests in a way that can automatically be interpreted by software applications. In an alternative operation mode, it pushes the composed service to the service library.