A framework for interoperability of BPEL-based workflows[†①]

Li Xitong (李喜彤) ②, Fan Yushun, Huang Shuangxi
(Department of Automation, Tsinghua University, Beijing 100084, P. R. China)

Abstract

With the prevalence of Service-oriented Architecture (SOA), web services have become the dominating technology to construct workflow systems. As a workflow is the composition of a series of interrelated web services which realize its activities, the interoperability of workflows can be treated as the composition of web services. To address it, a framework for interoperability of BPEL-based workflows is presented, which can perform three phases, that is, transformation, conformance test and execution. The core components of the framework are proposed, especially how these components promote interoperability. In particular, dynamic binding and re-composition of workflows in terms of web service testing are presented. Besides that, an example of B2B collaboration is provided to illustrate how to perform composition and conformance test. Finally, conclusions and future works are draw up.

Key words: interoperability of workflows, colored petri nets, conformance test, web service testing

摘　要

随着面向服务体系架构的日益流行，Web服务已经成为构建工作流系统的一项主流技术。一个工作流是由一系列实现流程中不同活动的相关Web服务组合而成，因此工作流的互操作可以看作是这些Web服务的组合。为了更好的实现工作流之间的互操作，提出了一个基于业务过程执行语言（BPEL）的工作流的互操作框架。.该框架能够执行三个阶段，即：转换、互操作检测、执行。介绍了各个核心模块的功能。特别地，该框架引入了基于Web服务测试方法的动态绑定和工作流重组合技术。通过一个B2B协同业务实例展示了如何进行工作流组合以及互操作检测。最后，总结全文并给出进一步工作。

0　Introduction

Nowadays, the realization of an enterprise's business goal requires highly efficiency of collaborative processes with its partners. To achieve it, the integration of inter-enterprise processes and interoperability of corresponding workflows have been paid much attention. Recently, web services have become the dominating technology to construct workflow systems. In such an environment, a workflow is considered to be the composition of a series of interrelated web services which realize its activities. And the workflow itself can be encapsulated as a composite service. Therefore, the interoperability of workflows can be treated as the composition of services which are usually documented with BPEL. Web services, however, are not always exactly compatible and much effort has to be ad-

① Supported by the High Technology Research and Development Programme of China (No. 2006AA04Z151 and 2006AA04Z166), the National Natural Science Foundation of China (No. 60674080 and No. 60504030), and IMPORTNET funded by the EU under FP6 in the area of ICT for Enterprise Networking (No. 033610).

② To whom correspondence should be addressed. E-mail: lxt04@mails.tsinghua.edu.cn

Received on

dressed to verify these pre-produced but incompatible services. By promoting correct compositions, conformance test has been a definite working area in the field of software engineering in recent years [1, 2].

In this paper, a framework for interoperability of BPEL-based workflows is presented, which can perform three phases, that is, *transformation*, *conformance test* and *execution*. In the first phase, web services specified in BPEL files are translated to Colored Petri nets (CPNs) that are described in Petri Net Markup Language (PNML) [3]. Then in the phase of conformance test, services are composed and composite workflow models are performed on a CPN simulator, that is, Petri Net Kernel (PNK) [4]. Any non-conformance will be checked and some test suites are generated. Finally, only valid composite service-based workflows are executed in distributed workflow systems. To promote interoperability with external services, dynamic binding and re-composition of workflows in terms of web service testing are presented.

The rest of the paper is structured as follows. Related work is presented in the next section. In Section 2, the framework is proposed along with three phases. And in Section 3, web service testing for dynamic binding is introduced. In Section 4, an example of Business-to-Business (B2B) collaboration is provided to illustrate how to use the framework. Finally, conclusions and the future work are drawn up in Section 5.

1 Related work

Recently, existing literatures about integration and interoperability of workflows mainly focus on how to utilize technology of web services. A business processes integration and monitoring system is developed in [5] to automate, integrate and monitor web processes. An architectural approach to the Business Process Integration (BPI) that highlights the ideal integration methodologies for applications is introduced in [6]. And a few approaches for EAI are developed based on combination of several technologies. An integration model is proposed in [7] for the combination of semantic web services, workflows and ontology. In [8], a reconfigurable web service integration system is developed in an extended logistics enterprise. Few approaches, however, are investigated to deal with dynamic binding and re-composition of workflows in the service-oriented computing environment.

The peculiar characteristics of SOA raise serious integration testing issues [9]. Since traditional testing methods don't work with web services, web service testing is a very newly and immature discipline in intense need of further research [10]. In [11], a method based on the design-by-contract testing technology is proposed to automatically generate test data for web services. And to quantify the trustworthiness of web services, an approach based on fuzzy c-mean algorithm is developed in [12]. However, there are very few researches regarding the use of web service testing for interoperability. In [13], a framework that extends UDDI registry is proposed, which can validate service behavior before actually registering it and facilitate the coordination among services registered at the same UDDI. But the framework does not focus on the phase of binding or execution. To address it, a framework based on the technology of conformance test to facilitate dynamic binding and interoperability of workflows is developed in the paper. Since web services are usually specified in BPEL files, the mechanism for transformation BPEL-based services to Petri net models has been investigated in the existing works [14, 15], which is utilized by our framework.

2 Framework for interoperability of BPEL-based workflows

In order to address logic conformance of processes and dynamic binding, a framework consisting of three core components is presented, as shown in Fig. 1. The core components are Web Service Transformer (WS-Transformer), Web Service Tester (WS-Tester) and Distributed Local Workflow Management System (DL-WfMS), which are respectively responsible for three phases, that is, transformation, conformance test and execution. In the first phase, after acquired from local UDDI broker, web services specified in BPEL files are translated to Colored Petri nets (CPNs) that are described in PNML formats. Then in the phase of conformance test, services are composed and composite workflow models are performed on Petri Net Kernel (PNK) which is a CPN simulator. Any non-conformance will be checked and some test suites can be generated. Only services with conformant logics can be imported into the execution environment of DL-WfMS. During execution, the processes will be monitored and invalid part of the workflow can be dynamically replaced. And alternative services can be fetched from local or public UDDI broker and rebound. In the following sections, the three phases are introduced in detail.

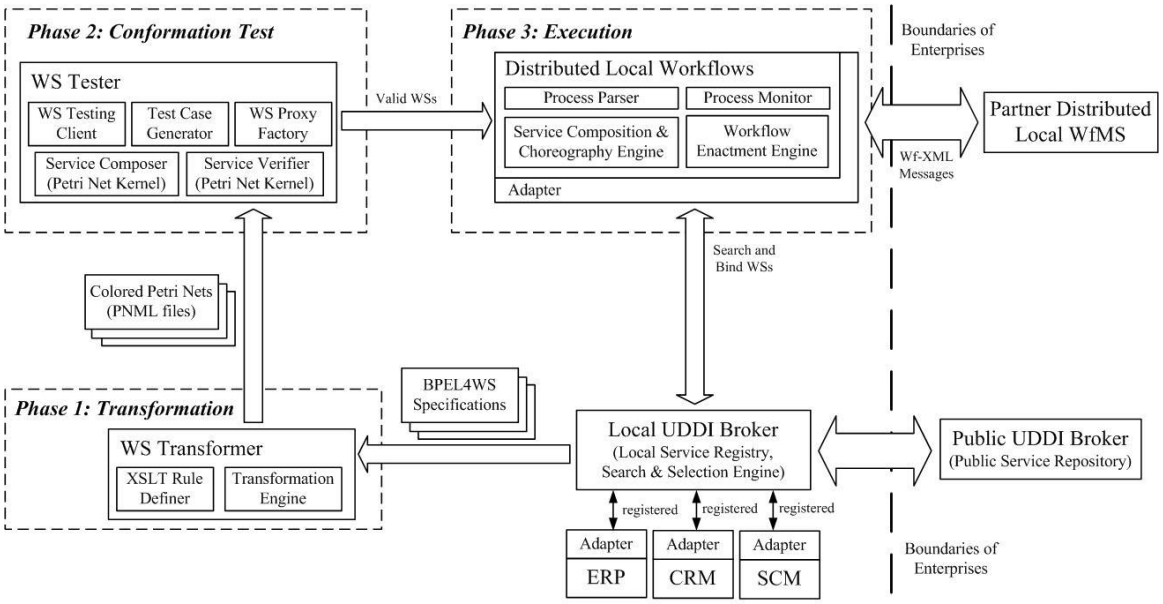

Fig. 1 Framework for interoperability of workflows

2.1 Phase 1: transforming BPEL specifications to colored petri nets

WS-Transformer has two subcomponents: one is XSLT Rule Definer and the other is Transformation Engine. In our framework, Petri nets are specified in PNML format [3] which is a proposal of an XML-based interchanging format for Petri nets and gaining increasing support from many Petri net tools. And transforming rules are defined using XSLT Rule Definer. Expressing how the basic constructs of BPEL are transformed to corresponding basic structural patterns of CPNs, the transforming rules are defined in XSLT files. XSLT, as a pattern-based language to transform a XML file in one format to another, is exactly proper for the transformation from BPEL to PNML. Transformation Engine is actually an XSLT processor, like XT, LotusXSL, SAXON, and its function comprises not only the basic transformation but refinement operations, such as combination places/transitions and reduction the size of a net as much as possible (by removing transient states).

```
<invoke partner="WS1" operation="ReqStkQuote" inputVariable="StkId" outputVariable="StkQuote" />
```

Fig. 2 A simple operation <*invoke*> of BPEL

```
<place id="i"><name><text> StkId </text></name></place>
<place id="o"><name><text> StkQuote </text></name></place>
<transition id="op"><name><text> ReqStkQuote </text></name></transition>
<arc id="aiop" source="i" target="op">…</arc>
<arc id="aopo" source="op" target="o">…</arc>
```

Fig. 3 A corresponding fragment of PNML file

```
<xsl:element name="place">
    <xsl:attribute name="id">i<xsl:number value="position()" format="1"/></xsl:attribute>
    <xsl:element name="name">
        <xsl:element name="text"><xsl:value-of select="@inputVariable"/></xsl:attribute>
    </xsl:element>
</xsl:element>
<xsl:element name="place">
    <xsl:attribute name="id">o<xsl:number value="position()" format="1"/></xsl:attribute>
    <xsl:element name="name">
        <xsl:element name="text"><xsl:value-of select="@outputVariable"/></xsl:attribute>
    </xsl:element>
</xsl:element>
<xsl:element name="transition">
    <xsl:attribute name="id">op<xsl:number value="position()" format="1"/></xsl:attribute>
    <xsl:element name="name">
        <xsl:element name="text"><xsl:value-of select="@operation"/></xsl:attribute>
    </xsl:element>
</xsl:element>
<xsl:element name="arc">
    <xsl:attribute name="id">aiop<xsl:number value="position()" format="1"/></xsl:attribute>
    <xsl:attribute name="source">i<xsl:number value="position()" format="1"/></xsl:attribute>
 <xsl:attribute name="target">op<xsl:number value="position()" format="1"/></xsl:attribute>
</xsl:element>
<xsl:element name="arc">
    <xsl:attribute name="id">aopo<xsl:number value="position()" format="1"/></xsl:attribute>
    <xsl:attribute name="source">op<xsl:number value="position()" format="1"/></xsl:attribute>
 <xsl:attribute name="target">o<xsl:number value="position()" format="1"/></xsl:attribute>
</xsl:element>
```

Fig. 4 A certain transforming rule specified with XSLT

To describe transforming rules using XSLT in detail, we take a simple operation of BPEL, that is <*invoke*>, for example. As shown in Fig. 2, a certain service requests its partner *WS1* for some stock information in terms of its operation *ReqStkQuote*. The input and output messages of the operation are *StkId*, *StkQuote*. The definition of BPEL-based operation can be transformed to a fragment of PNML file that depicts a Petri net model, as shown in Fig. 3, by using a certain transforming rule. The transforming rule specified with XSLT is presented in Fig. 4. Note that basic constructs of BPEL, like sequence, parallel, exclusive choice and iteration, can also be transformed to PNML fragments in terms of corresponding patterns specified with XSLT, which is not presented in the paper, due to the space limitation.

2.2 Phase 2: conformance test of composite workflow model

As responsible for conformance test, WS-Tester is a significant component in the framework. Firstly, CPN-based service models are imported to composer which is a subcomponent of WS-Tester. In the composer, two or more CPNs are composed together to a single composite net by combining message places. Then a service verifier will perform verification on the composite net, especially checking properties of deadlock-freedom. In our framework, service composer and verifier are both developed on the basis of Petri Net Kernel (PNK) [4] which is a Petri net tool and provides infrastructure for analysis, simulation and verification. When performed on PNK, a reachability tree of the composite net is constructed. Each leaf of the reachability tree is checked whether it is one of the final states or not. If not, the leaf represents an un-terminating state and the trace from the root to the leaf is considered as an un-completing trace, which means that some deadlock will occur in certain situation and the composite net is not dead-lock-free. As long as each leaf is a final state, the trace from the root to the leaf is a successful one and the composite net are a deadlock-free net. In this case, the composite workflow is a logic-conformant workflow. The other three subcomponents, like WS Proxy Factory, Test Case Generator and Testing Client, will be used by web service testing and introduced in the following section for dynamic binding.

2.3 Phase 3: execution of distributed local workflows

There are four parts in the distributed local WfMS. Similar to traditional WfMS, the process parser provides the ability of parsing process definition files for the workflow enactment engine, while the process monitor can monitor status of process execution and collect data generated during the execution. When some failure occurs or its non-functional properties go down, the process monitor will send alarms and take some actions. In an on-demand and fast changing environment, dynamic binding can provide more flexibility and scalability to the application systems. In this section, we refer to the process of automatically binding web services to abstract process definitions.

To dynamically compose and bind web services of a workflow, a service composition and choreography engine has been introduced in the DL-WfMS. The engine is responsible for the composition and choreography of web services which should be correctly tested by the testing module. Moreover, it can provide the ability of re-composition and dynamic binding, which is an attractive characteristic introduced by our proposed framework. Since BPEL has been approved as an OASIS standard for web services and many open source workflow engines, like ActiveBPEL [16], Twister [17], etc., support the execution of BPEL-based services, the engine presented herein uses BPEL-based service models for composition and choreography. It should be pointed out that composition and choreography are two different functionalities of the engine. The main difference lies in whether a dominating service exists or not.

When a dominating service exists and other services interact with it as partners, the engine composes other partners with the dominating services. Otherwise, all services are equally treated and the engine choreographs the entire workflow from a global view.

When some functional operation of a web service fails or its quality depresses at runtime, the service composition and choreography engine will perform the following steps to achieve re-composition and dynamic binding:

1. Suspend the running workflow;
2. Store the execution status in a buffer;
3. Select alternative service that provides similar functional operations and non-functional properties;
4. Retrieve the status information and put it into the new service;
5. Switch the executive entity of the abstract process definition to the new service;
6. Restart the execution of the workflow.

Besides that, adapter is a significant concept in our framework, especially for interoperability of workflows. Almost every application system needs to be integrated and requires adapters. When incompatibilities of protocols of two services are identified, a few of existing approaches can be used to automatically develop adapters that can glue them together [15].

3 Dynamic binding for interoperability

As mentioned above, in an on-demand and fast changing environment, dynamic binding can provide more flexibility and scalability to application systems. There are a couple of publications regarding dynamics of web services [18, 19]. However, few of them refer to the means of web service testing in dealing with dynamics and interoperability of workflows, which is one of the contributions of the paper. Herein, we present an approach to dynamic binding and re-composition of workflows by means of web service testing.

Web service testing plays an important role in our proposed framework, with which the framework can achieve the capabilities of testing and dynamic binding of web services. Fig. 5 depicts two loops: the upper loop supports dynamic binding of workflows, namely run-time binding, while the lower loop illustrates the main steps referring to web service testing. We present the explanation as follows:

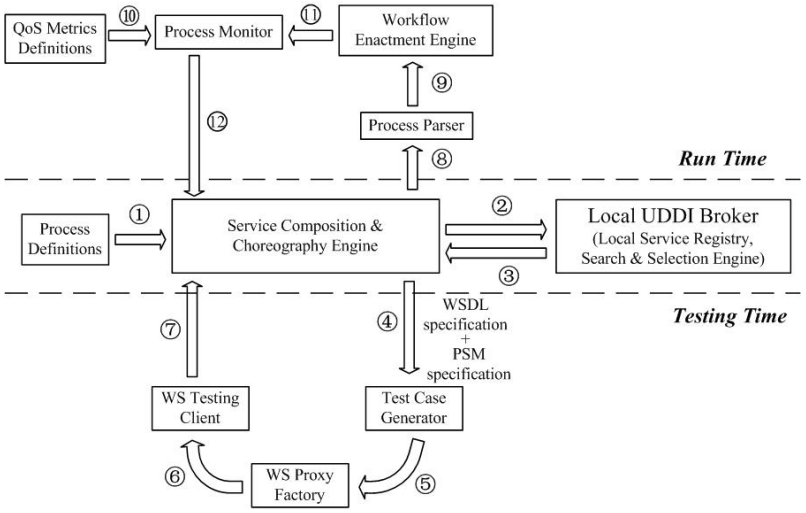

Fig. 5 Steps of web service testing and dynamic binding

1. The process definition tool sends the abstract process definition files, such as BPEL files, to the service composition and choreography engine. Containing process models and sequences of involved activities, the abstract process definition files exclude specific binding information of the activities;
2. According to process definition files, service composition and choreography engine invocate local UDDI broker and request for executive entities of activities;
3. By means of service discovery algorithms, the local UDDI broker searches web services satisfying the functional and non-functional requirements and returns binding information of these services;
4. The service composition and choreography engine send the WSDL and Protocol State Machine (PSM) specifications of the selected web services to the test case generator;
5. The test case generator produces test cases in terms of WSDL and service process models. Existing works have provided a few technologies to generate test cases [9, 20]. For functional testing, mutation strategies play an important role. Mutation strategies change or mutate the input SOAL messages of a web service and analyze the service responses to check whether these mutations produce observable effects in the service outputs. On the other hand, search-based stress test can be used for non-functional testing;
6. By using the generated test cases, WS proxy factory generates WS proxy for the tested service. The WS proxy may be invocated by the tested web service as its partner. Practically, a WS proxy is not a real web service but several invocating and receiving operations which are specified in SOAP messages;
7. Some partner of the tested web service can be obtained in UDDI registry, which acts as a client for testing. The WS testing client as well as involved WS proxy invocate the service composition and choreography engine and a workflow instance for testing will be started;
8. When selected web services have been tested to be usable, the service composition and choreography engine will bind the specific services to process definition files and deliver to the process parser;
9. The process parser parses the definition file and sends it to workflow enactment engine;
10. Several metrics about quality of services (QoS) are put into the process monitor;
11. The process monitor monitors the status of execution and collecting data generated during the execution;
12. The process monitor delivers the status of the process execution to the service composition and choreography engine. According to some predefined criteria, the service composition and choreography engine estimates whether the status of process execution is satisfied. If some web service should be replaced for higher performance, the service composition and choreography engine will invocate local UDDI broker and request for another similar web service, which returns to the second step above.

What we emphasize herein is that the upper loop of dynamic binding and lower loop of web service testing are both closed loops and can execute dependently. If a running web service should be replaced, the service composition and choreography engine will start the lower testing loop. And if the tested web service fails to match a specific functional or non-functional requirement, the engine can give up binding the tested web service and request local UDDI Broker for another one. And then the web service testing loop recycles again.

4 An example of business-to-business collaboration

Herein we present a Business-to-Business (B2B) collaboration of online ordering goods, within which the seller and customer are both web services specified in BPEL files. Due to the length limitation, BPEL specifications of seller and customer services are not given. For the sake of illustration, the orchestration processes of the two inte-

racting services are presented by using activity diagrams, as shown in Fig. 6. Seller stays in the state of idle until customer service sends a purchase order. After receiving order, seller checks inventory whether it is available or not. If unavailable, seller sends a message of no goods to customer and return to the state of idle. And after receiving message of no goods, customer changes to another seller and the collaboration process is terminated. If inventory available, seller requests customer for money. In this case, customer pays money and waits for goods. After receiving money, seller delivers goods to customer and return to the state of idle. And after receiving goods, customer's process is completed. This is a simplified scenario and there are only four message exchanges between the seller and customer. But it is sufficient to illustrate interactions within a collaboration process and interoperability of workflows of the seller and customer.

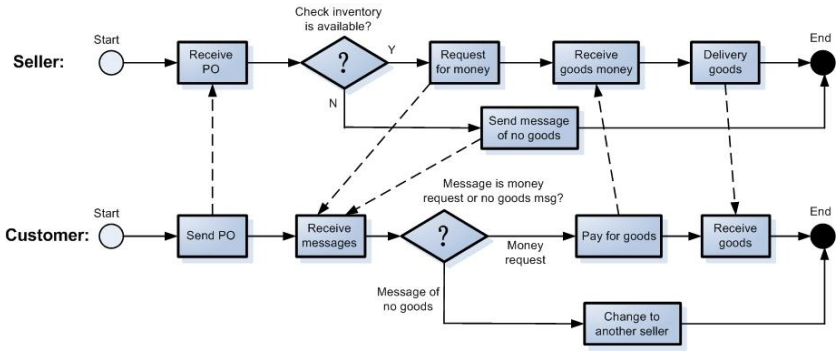

Fig. 6 A B2B collaboration process of online purchasing goods

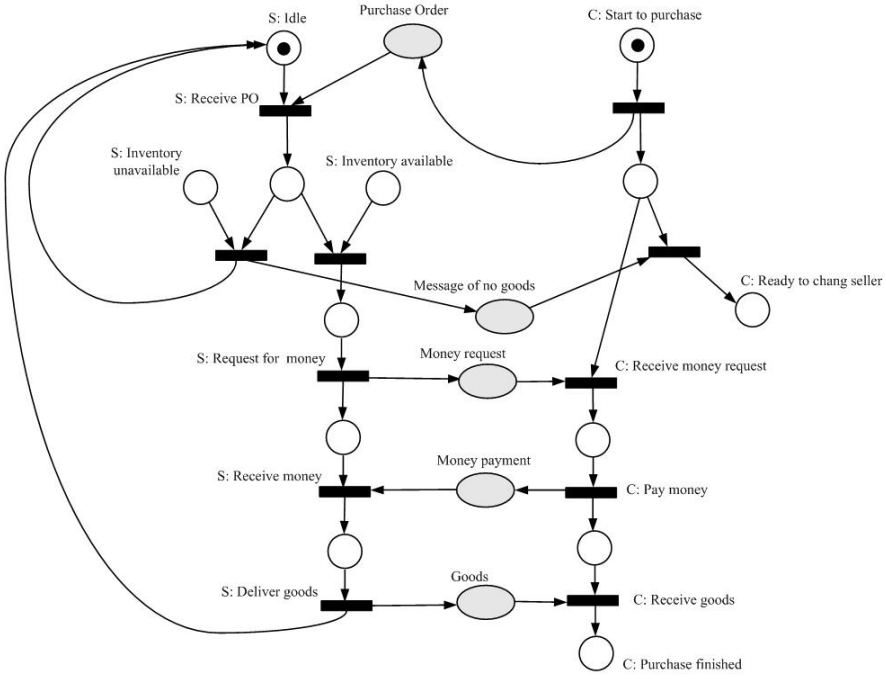

Fig. 7 Composite petri net of the collaboration process

By using WS-Transformer of the framework, the BPEL specifications of seller and customer are transformed to two Colored Petri nets. And then, the two CPNs are imported to Service Composer. In Service Composer, common message places are combined together and the two CPNs are composed into a composite CPN, as shown in Fig. 7. The message places are presented with gray ellipse, through which seller and customer interact with each other.

When performing the composite CPN on Service Verifier, a snapshot is presented, as shown in Fig. 8. With the help of Service Verifier that is based on Petri Net Kernel (PNK), the reachability tree is analyzed and the analysis is based on tree branch coverage. All possible traces are searched and non-conformance can be detected. In this simplified example, there is no non-conformance and only two traces are identified. Based on the identified traces, two test suites are generated, that is, TestSuite_1 = (C: start to purchase, S: inventory unavailable) and TestSuite_2 = (C: start to purchase, S: inventory available).

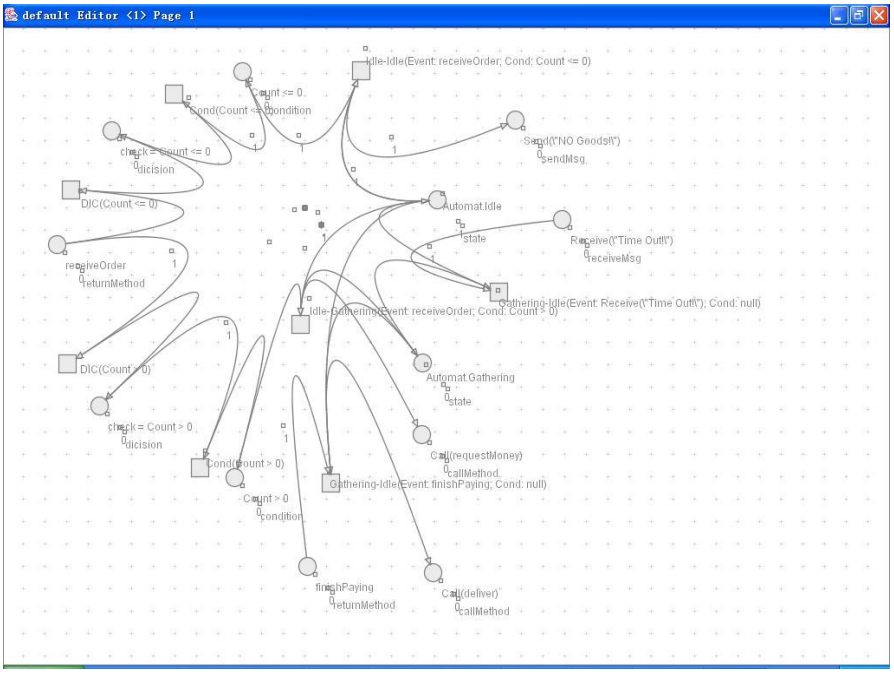

Fig. 8 Snapshot of verification on petri net kernel

5 Conclusions and future work

With the prevalence of Service-oriented Architecture, web services have become the dominating technology to construct workflow systems. And the interoperability of workflows can be treated as the composition of web services. In this paper, a framework for interoperability of BPEL-based workflows is presented, which can perform three phases, that is, transformation, conformance test and execution. The core components and how these components promote interoperability of workflows are presented. Particularly, dynamic binding and re-composition of workflows in terms of web service testing are introduced. In addition, an example is provided to illustrate how to perform composition and conformance test.

In the future, we will carry out further researches to explore related technologies in the proposed framework and to develop methods supporting run-time and test-time automatically executions of dynamic binding. Besides that, we plan to improve the prototype system to verify the feasibility and efficiency of the proposed framework.

References

[1] Baldoni M, Baroglio C, Martelli A, et al. A priori conformance verification for guaranteeing interoperability in open environments. In: Proceedings of International Conference on Service-Oriented Computing, 2006, 339-351.

[2] Wei Y X, Zhang S S, Zhong F R. On formalizing and verifying web services. Journal of High Technology Letters, 2005, 11(1):47-50.

[3] Jonathan B, et al. Petri net markup language: concepts, technology, and tools. ICATPN, 2003.

[4] Petri Net Kernel Team. Petri net kernel, http://www.informatik.hu-berlin.de/top/pnk/, 2002.

[5] Hernandez G, Hernandez C, Aguirre J. BPIMS-WS: brokering architecture for business processes integration in B2B e-commerce. In: Proceedings of the 15th International Conference on Electronics, Communications and Computers, 2005, 160-165.

[6] Raut A, Basavaraja A. Enterprise business process integration. In: Proceedings of Conference on Convergent Technologies for Asia-Pacific Region, 2003, 1549-1553.

[7] Alexakos C, Kalogeras A, Likothanassis S, et al. Workflow - coordinated integration of enterprise. In: Proceedings of the 10th IEEE Conference on Emerging Technologies and Factory Automation, 2005, 273-279.

[8] Talevski A, Chang E, Dillon T. Reconfigurable web service integration in the extended logistics enterprise. IEEE Transaction on Industrial Informatics, 2005, 1(2):74-84.

[9] Canfora G, Di Penta M. Testing services and service-centric systems challenges and opportunities. Journal of IT Professional, IEEE Computer Society, 2006, 8(2):10-17.

[10] Tsai W, Chen Y, Paul R, et al. Adaptive testing, oracle generation, and test case ranking for web services. In: Proceedings of the 29th Annual International Conference on Computer Software and Applications, IEEE Computer Society, 2005, 101-106.

[11] Jiang Y, Xin G M, Shan J H, et al. A method of automated test data generation for web service. Chinese Journal of Computers, 2005, 28(4):568-577.

[12] Li X T, Fan Y S. An approach to web services testing based on fuzzy classification. Accepted by Journal of Chinese Computer Systems, 2007.

[13] Bertolino A, Polini A. The audition framework for testing web services interoperability. In: Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE Computer Society, 2005, 134-142.

[14] Ouyang C, Verbeek E, W.M.P. van der Aalst, et al. Formal semantics and analysis of control flow in WS-BPEL. Science of Computer Programming, 2007, 67(2-3):162-198.

[15] Tan W, Fan Y S, Zhou M C. A petri net-based method for compatibility analysis and composition of web services in business process execution language. Accepted by IEEE Transactions on Automation Science and Engineering, 2007.

[16] ActiveBPEL. http://www.active-endpoints.com/active-bpel-engine-overview.htm.

[17] Twister. http://www.smartcomps.org/twister.

[18] Zhu F, Turner M, Kotsiopoulos I, et al. Dynamic data integration using web services. In: Proceedings of International Conference on Web Services, IEEE Computer Society, 2004, 262-269.

[19] Zhou B, Tang J, He Z. An adaptive model of virtual enterprise based on dynamic web service composition. In: Proceedings of the 5th International Conference on Computer and Information Technology, 2005, 284-289.

[20] Dong W, Yu H, Zhang Y. Testing BPEL-based web service composition using high-level petri nets. In: Proceedings of 10th IEEE International Enterprise Distributed Object Computing Conference, 2006, 441-444.

Li Xitong, born in 1982. Now he is a Ph.D. candidate in Department of Automation, Tsinghua University. His research interests include workflows, web services, service-oriented computing and petri nets.