Control Flow Complexity Metrics for Petri Net-based Web Service Composition

Chengying Mao
School of Software and Communication Engineering,
Jiangxi University of Finance and Economics, 330013 Nanchang, China
Email: maochy@yeah.net

***Abstract*—Web services technology is an effort to build a distributed computing platform over the network, and it can implement systematic application-to-application interaction on the Web. In recent years, this new technology has been widely adopted for constructing distributed applications. However, how to precisely measure the controlling complexity of Web service composition (WSC) is a very difficult task due to its characters such as heterogeneity, distributed and loose-coupling. In the paper, we mainly concern on the complexity measurement of Petri net-based business process in Web service composition. Two metric sets are presented through analyzing the WSC's execution logics and dependency relations in workflow. The first one is count-based metric set, and includes seven metrics such as number of place, average degree of transition, transfer number per service and cyclomatic complexity. The second is an execution path-based metric set, which includes average execution path complexity (AEPC) and its extension based on cognitive informatics. In addition, two real-world WSCs are used to validate our measurement methods. The results show that our metrics are effective and rational, and have high practical value for WSC analysis and maintenance.**

***Index Terms*—complexity analysis, Petri net, Web service composition, execution path, cognitive informatics**

I. INTRODUCTION

With the rapid development of network technology, distributed computing has become the important and mainstreaming pattern for designing and executing software system. Compared with the traditional techniques such as CORBA, RPC and DCOM, service-oriented architecture (SOA [1]) provide better interoperability for data exchange and application invocation. In this new software development pattern, Web service [2,3] is the typical technique and puts this new idea into practice.

Web services technology [3] is an effort to build a distributed computing platform over the network, and can implement systematic application-to-application interaction on the Web. Although it can bring lots of benefits for building a flexible and open-accessing software system, the related problems of system comprehension, testing and maintenance are still open issues. Due to the characters of Web service composition (WSC), such as heterogeneity, distributed and loose-coupling, how to measure system's complexity is a challenging task in the research community of software measurement [4].

Web service components don't work disorderly in WSC, contrarily they are well regulated according to the system business process. In general, such process is described in the form of workflow. Petri-Net [5,6] is a well-known model to represent the workflow both in business activities and computer systems. Of course, it also can be used to model the interaction relations between Web services. Until today, Petri net-based business activity modeling has been explored by some researchers [7,8], and has become one of important process representation techniques in WSC. In this paper, some complexity metrics for Web services workflow described by Petri-Net will be proposed. At first, we analyze the basic elements of business process and the corresponding Petri-Net representations. Then, the metrics about information flow (especially control flow) in WSC will be addressed. In addition, in order to validate the feasibility and effectiveness of our proposed metrics, two real-world Web service compositions are used as a subject system in our case studies. The analysis results show that our metrics can reasonably reflect the complexity feature of Web service-based system.

The remainder of this paper is organized as follows. In Section 2, we analyze the basic characters of Web service composition. Meanwhile, a running-example service composition is introduced to demonstrate the following measurements. The metrics about information flow in Web service composition, i.e., count-based metrics and execution path-based metrics, are presented in Section 3 and 4 respectively. In Section 5, two cases about real-world Web service compositions are studied to confirm the effectiveness of our proposed metrics. The related work is addressed in Section 6, and Section 7 concludes the paper.

II. BACKGROUND

In this section, we firstly review the basic features of Web service composition, and then give the atomic Petri-Net model for the basic composition logic. In order to address our complexity measurement methods, a service composition example is introduced here.

A. Basic Logics in WSC

In fact, Web services technology provides a way to integrate some distributed service units over the network

doi:10.4304/jsw.5.11.1292-1299

into a coordinative system. To ensure such services can correctly work together, the whole system should be run under the constraint of its specific business process. In general, the business process is a workflow which can be represented in the form of Petri-Net, BPEL [9] or BPMN [10]. In this paper, we mainly concern on the Petri net-based representation. Here, let us review the Petri-Net definition [6,7,11] at first.

Definition 1 (*Petri-Net*). A Petri net is a triple $PN = (P, T, F)$, where P is a finite set of places, T is a finite set of transitions representing the operations, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs. Tokens are contained in the places, and the distributions of tokens reflect different system statuses.

In the Web service-based system, the activities in the corresponding business process can be classified into two types: basic activities and structure activities. The basic activity is atomic, such as `receive`, `reply`, `invoke`, `assign`, `throw` and `exit`. The structure activity includes `sequence`, `if`, `while`, `repeatuntil`, `pick` and `flow`. It should be noted that, the flow (`<flow>`) activity provides concurrency and synchronization, and is used to define a set of activities that will be invoked in parallel. In order to represent the structure activities in business process in WSC, the following four basic logic models should be adopted.

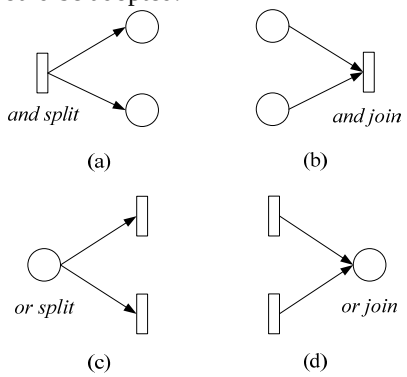

Figure 1. Four basic logic models for structure activity representation

It's not hard to find that, structure activities in WSC can be summarized into the following four categories: sequence, branch, loop and parallel. Based on the above four basic logic models, the Petri-Net representations of four basic activities can be illustrated in Figure 2.

Although the business process in real executing scenarios is very complex, it can be represented by the above four basic activity structures in the way of nested composition. While comparing Web service composition specifications with Petri-Net notation, it is obviously that the transitions in Petri-Net model represent the operations in business process, such as value assignment, message reply and service invocation. On the other hand, the place in Petri-Net can be condition judgment or connector between two operations.

B.Running Example

In order to describe our complexity measurement methods for Web service composition, an `OnlineOrder` example [11] is introduced here. It is a typical business to business (B2B) application, and its business process is described in Figure 3. This application firstly receives an order form, then checks it and queries the credit record of the ordering customer. If the customer has a good credit record and the order form is approved, system will perform production planning and arrange product freight. Otherwise, the order form will be rejected directly.

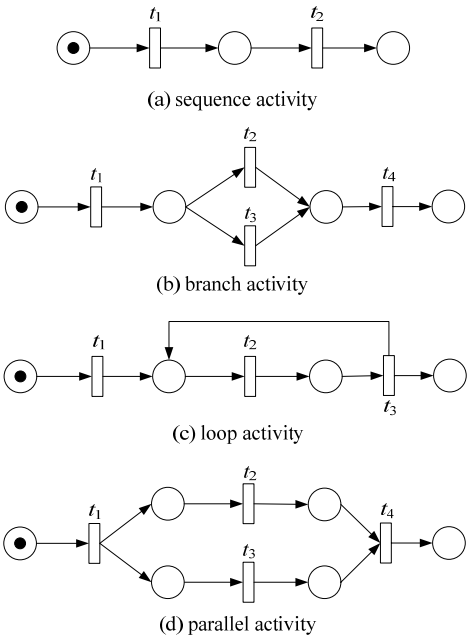

Figure 2. Petri-Nets for the four basic activities in WSC

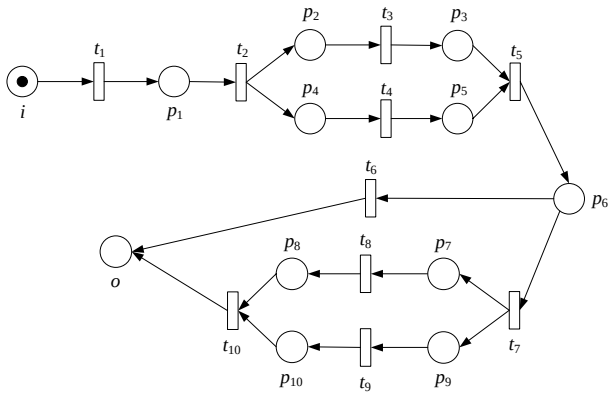

Figure 3. Petri net-based business process of OnlineOrder application

In the above business process, the corresponding Petri net has 12 places (including the start place i and the end place o) and 10 transitions. The meanings of these components are addressed in the following table. It is noteworthy that the places which are not explained in Table 1 have no specific function and are merely for the connection purpose.

TABLE I. MEANINGS OF THE SYMBOLS IN ONLINEORDER PETRI NET

Symbol	Operation	Comment
t_1	receive order information	basic activity (receive)
t_2	<flow>	flow activity

t_3	credit checking	service invoking
t_4	order checking	service invoking
t_5	</flow>	end of flow
p_6	if condition	branch activity
t_6	reject order	basic activity (reply)
t_7	<flow>	flow activity
t_8	production planning	service invoking
t_9	freight arrangement	service invoking
t_{10}	</flow>	end of flow
o	end of if condition	join node

According to the above description, we can know that four Web services are invoked in this Web service composition, which are denoted by transition t_3, t_4, t_8 and t_9, respectively. In addition, there are two parallel execution bodies (i.e., $t_2 - t_5$ and $t_7 - t_{10}$) and one branch sub-structure (i.e., $p_6 - o$) in the business process of this WSC.

In this paper, we will provide two kinds of complexity metrics, i.e., count-based metric and execution path-based metric. The former is the issues reflecting static features of business process and the latter is dynamic complexity metric.

III. COUNT-BASED MEASUREMENT

Count-based metric is the most naïve approach to scale the complex degree of constructs in program, process or network. In this paper, we adopt it as a basic method to measure the structure complexity of business process represented by Petri-Net in Web service composition.

(1) Number of Places

This issue is the total number of places in Petri net-based business process, and it reflects the data exchange times in whole Web service composition. According to the definition 1, it can be easily measured by the following formula.

$N_P = |P|$, where P is the place set in Petri net. (1)

It is not hard to find that, the larger value of N_P, the more frequent data storage, transfer or exchange will appear in the business process of WSC. As mentioned in Section 2.B, $N_P = 12$ in the example application.

(2) Number of Transitions

In Petri net-based business process, a transition usually stands for an operation or parallel control logic in Web service composition. Hence, the number of transitions can reflect the activity number in system. Similarly, it also can be calculated as shown in formula (2).

$N_T = |T|$, where T is the transition set in Petri net. (2)

In general, if a service process contains more activities, it must be more complex than the process with fewer activities. Therefore, the Petri net-based process with higher N_T means higher structure complexity. Obviously, $N_T = 10$ in the `OnlineOrder` B2B system.

(3) Number of Services

In a Web service-based system, the number of invoked services directly reflects interaction complexity with external system. This item can be expressed as follows.

$N_S = |S| = |\{s_i\}|$ (3)

Where $S \subset T$ is invoked service set in Web service composition, and s_i refers the specific service used in the service invocation site in the Petri net-based business process.

Generally speaking, if a Web service composition involves more service units, it means that such application has more frequent interaction with external services supplied by service providers. While consider the running example application, there are four external services in its business process. Thus, $N_S = |\{s_i\}| = |\{\text{credit checking, order checking, production planning, freight arrangement}\}| = 4$.

(4) Average Degree of Place

From the perspective of network, the information of node degree in network reflects the interaction strength between nodes. In general, the more average degree means the higher interaction strength in network. There are two kinds of nodes in the Petri net-based business process, so we consider the interactions for place and transition respectively.

The average degree of place (ADP) can be computed via the following formula.

$$ADP = \frac{\sum_i deg(p_i)}{|P|} = \frac{\sum_i [indeg(p_i) + outdeg(p_i)]}{|P|} \quad (4)$$

Where $p_i \in P$ is the ith place in Petri net, and $deg(p_i)$ is the degree of node corresponding place p_i in network. It can be divided into two parts: $indeg(p_i)$ and $outdeg(p_i)$.

As mentioned above, place in the Petri net-based business process refers to data storage or branch judgment, so the item ADP can be used to reflect the data transfer complexity in WSC.

For the example application `OnlineOrder`, the value of ADP can be calculated as below.

$$ADP(OnlineOrder) = \frac{\sum_i deg(p_i)}{|P|} = \frac{24}{12} = 2$$

From the results we can find that, the data interaction in this application is not so complex. The current value means that each place has one input data port and one output port from the average sense.

(5) Average Degree of Transition

Another node in Petri net-based business process is transition node. Similarly, its average degree (ADT for short) can be yielded according to formula (5).

$$ADT = \frac{\sum_i deg(t_i)}{|T|} = \frac{\sum_i [indeg(t_i) + outdeg(t_i)]}{|T|} \quad (5)$$

Where $t_i \in T$ is the ith transition in Petri net, and $deg(t_i)$ is the node degree of transition t_i in network.

It is not hard to find that, the average degree of transition can reflect the parallel complexity of business process, i.e., the parallel execution ability of Web services in application. The value of this issue for the

example Web service composition can be computed as follows.

$$ADT(OnlineOrder) = \frac{\sum_i deg(t_i)}{|T|} = \frac{24}{10} = 2.4$$

Since the *ADT* value is greater than 2, it means that the example business process has parallel execution ability for service units, that is, some parallel execution bodies should exist in the process.

(6) Transfer Number per Service

Edges in Petri net-based business process represents the data and control logic transfers in Web service-based system. Web service composition can be viewed as a collection of service units. Consequently, the number of transfers used to integrate a Web service into the system should be concern. The small number means the current WSC has good composition structure, otherwise not. So the item of transfer number per service (i.e. *TNS*) can be used as an indication of optimization degree of WSC's structure, and it can be calculated by formula (6).

$$TNS = \frac{|F|}{N_S} = \frac{\sum_i [deg(p_i) + deg(t_i)]/2}{N_S} \quad (6)$$

Where F is a set of directed arcs in Petri net.

For the `OnlineOrder` example, there are four external services are integrated into system, and the number of arcs in Petri net-based business process is 24. As a consequence, the *TNS* value of this example system can be expressed as

$$TNS(OnlineOrder) = \frac{|F|}{N_S} = \frac{24}{4} = 6.$$

(7) Cyclomatic Complexity

McCabe cyclomatic complexity (*CC*) [12] is one of the most widely used software metrics. It is also suitable for weight the structure complexity of business process in Web service-based system [13]. For the business process denoted by Petri net, its cyclomatic complexity can be calculated as below.

$$CC = |F| - |P| - |T| + 2 \quad (7)$$

Similar to the traditional program, the value of this item reflects the complexity of control structure in business process. The higher value means more complex control relation between Web services. For the running example, its *CC* value is $24 - 12 - 10 + 2 = 4$.

IV. Execution Path-based Measurement

Count-based measurement can only reflect the static feature of Web service composition, so it needs another way to describe the dynamic character of Web services. In this section, we will address execution path-based metrics to scale the dynamic execution complexity of WSC. At first, the basic execution path-based metric is introduced. Then, an extension is proposed through adopting the knowledge of cognitive informatics.

A. Basic Execution Path-based Metric

During the execution of Web service-based system, the computing time is determined by the execution path in the current scenario. On the other hand, all possible execution paths should be considered when a maintainer attempt to understand such system. Therefore, the complexity of execution path can be used as an indication of dynamic execution behaviors of Web service-based system.

Generally speaking, WSC is not great different from the traditional program. The significant difference lies in the parallel structure in the business process of WSC. In order to identify the execution paths in WSC, the concept of parallel execution relation is defined firstly. In the following context, the *place* and *transition* in Petri net-based business process is uniformly referred to as *node*.

Definition 2 (*Parallel Execution Relation*). Suppose seq_1 and seq_2 are two sequences immediately following an "*and split*" transition node in Petri net-based business process, these sequence will execute at the same time when the WSC is running in some specific scenario, denoted as $seq_1 \| seq_2$. Meanwhile, the nodes (e.g., n_1 and n_2) in parallel sequences also have such relation in Petri net, i.e., $n_1 \| n_2$.

For example, the sequence $p_2 \to t_3 \to p_3$ has the parallel execution relation with $p_4 \to t_4 \to p_5$ in Figure 3. Similarly, the sequence $p_7 \to t_8 \to p_8$ and $p_9 \to t_9 \to p_{10}$ also have the parallel execution relation. Based on the above definition, we can introduce a concept of execution path here.

Definition 3 (*Execution Path*). In the execution of Web service composition, the sequence composed of *place* nodes and *transition* nodes is called *execution path*. Obviously, an execution path perhaps contains serial sub-sequences and parallel sub-sequences.

For the serial sub-sequence, we can denote it in the form of $n_i \to n_j \to \cdots \to n_k$. By contrast, the parallel execution sub-sequences in execution path can be expressed in the following form: $(n_u \to \cdots \to n_v, \cdots, n_x \to \cdots \to n_y)$. Accordingly, an execution path can be formed by nested combination of such sub-sequences.

Take the `OnlineOrder` application for an example, there are two execution paths in its business process. According to the notation provided in the above definitions, these paths can be expressed as below.

$$Path1 = i \to t_1 \to p_1 \to t_2 \to (p_2 \to t_3 \to p_3,\ p_4 \to t_4 \to p_5) \to t_5 \to p_6 \to t_6 \to o$$

$$Path2 = i \to t_1 \to p_1 \to t_2 \to (p_2 \to t_3 \to p_3,\ p_4 \to t_4 \to p_5) \to t_5 \to p_6 \to t_7 \to (p_7 \to t_8 \to p_8,\ p_9 \to t_9 \to p_{10}) \to t_{10} \to o$$

In this section, we evaluate the complexity of whole Web service composition through analyzing the execution path complexity. For the above execution paths, their complexities can be defined in the following style.

Definition 4 (*Execution Path Complexity*). Given an execution path *Pt* , the control complexity of *Pt* can be defined as the sum of complexities of all places and transitions in this execution path. Formally,

$$C(Pt) = \sum_i C(p_i) + \sum_j C(t_j) \quad (8)$$

Where p_i is the place node in path *Pt* , and t_j is transition node in this path.

For the purpose of simplicity, the complexity of each place or transition node is assigned with the weight 1, i.e., $C(p_i)=1$ and $C(t_j)=1$. Hence, the complexity of an execution path can viewed as the node (including place and transition) number in the path. While considering the above two example paths, their complexities are 14 and 21 respectively.

After getting complexities of all execution paths, the dynamic complexity of the corresponding WSC can be scaled by average execution path complexity (AEPC). Given a Web service composition, the execution path set in it is denoted as $PS=\{Pt_1, Pt_2, \cdots, Pt_k\}$, and the execution probability of each path is denoted as $prob(Pt_i)$, then the AEPC can be measured by the following formula.

$$AEPC=\sum_{i=1}^{k} prob(Pt_i)\cdot C(Pt_i)$$
$$= prob(Pt_1)\cdot C(Pt_1)+\cdots+ prob(Pt_k)\cdot C(Pt_k) \quad (9)$$

It should be noted that, if the execution probability of each path is not addressed in system specifications, we can treat it in the basic manner, i.e., $prob(Pt_i)=1/k$, where k is the number of execution paths in WSC.

For the running B2B application, we suppose the probabilities of *Path*1 and *Path*2 are 0.3 and 0.7 respectively. Then, the AEPC value of whole application can be calculated as

$$AEPC(OnlineOrder)=0.3\times 14+0.7\times 21=18.9$$

From the result we can find that, the frequently used paths play more important role for weighting the dynamic execution complexity of whole Web service-based system. Therefore, the profile information is very useful for the complexity measurement of Web service composition.

B. Extension Based on Cognitive Informatics

In the above basic measurement, all place and transition nodes are assigned with the same weight. In fact, different place or transition node has different complexity for executing or understanding. For example, the "and split" and "and join" node are more complex than the common transition nodes in Petri net-based business process. Similarly, the "or split" and "or join" node are more complex than the place nodes for the general purpose of data transferring. Therefore, the above basic metric should be extended by assigning different types of nodes to different weights. In order to fulfill this task, an effective way is to measure the complexity of place or transition node from the perspective of cognitive informatics.

In cognitive informatics, it is found that the functional complexity of software in design and comprehension is dependent on internal architecture of the software [14]. The cognitive weight of software is the extent of difficulty or relative time and effort for understanding a given software control structure. Some previous researches in [14-16] provide the reference weights for the basic control structures. It is not hard to find that, there exists similar feature between the traditional program and Petri net-based business process from the perspective of cognitive comprehension. Here, we assign the complexity weights for the place and transition nodes in business process described by Petri net in the similar way.

TABLE II. COMPLEXITY WEIGHTS OF KEY STRUCTURE NODES

Type No.	Type Name	Basic Structure	Weight
1	Branch	1.1 or split (two-way)	2
		1.2 or split (many-way)	3
		1.3 or join (two-way)	2
		1.4 or join (many-way)	3
2	Iteration	2.1 while	3
		2.2 repeatUntil	3
		2.3 forEach	3
3	Concurrency	3.1 flow	4
		3.2 join node </flow>	4
4	Service Invocation	4.1 external service invoking	2
5	Interrupt	5.1 exception handler	3
		5.2 event handler	3

According to the above weight definitions, we can analyze the complexities of place nodes and transition nodes as follows. Place p_6 is an "or split" node and place o is an "or join" node, so the complexities of both them are 2, i.e., $C(p_6)=2$ and $C(o)=2$. Meanwhile, other place nodes are all basic ones, so their weights are all 1.

The complexity weights of transition nodes are more complicated here. Among them, t_2 and t_7 are "and split" nodes, thus $C(t_2)=C(t_7)=4$. Accordingly, t_5 and t_{10} are all "and join" nodes, so $C(t_5)=C(t_{10})=4$. Moreover, node t_3, t_4, t_8 and t_9 are not the common operations, but the service invocation nodes. Therefore, their complexities can be expressed as $C(t_3)=C(t_4)=C(t_7)=C(t_8)=2$.

Based on the above analysis for each node's complexity, the complexities of *Path*1 and *Path*2 can be calculated according to formula (8). Hence,

$C(Path1)=6\times 1+2\times 2+2\times 1+2\times 4+2\times 2=24$, and

$C(Path2)=10\times 1+2\times 2+1\times 1+4\times 4+4\times 2=39$.

Then, the average execution path complexity based on cognitive informatics (here denoted as $AEPC_{CI}$) can be computed as follows.

$$AEPC_{CI}(OnlineOrder)= prob(Path1)\cdot C(Path1)+$$
$$prob(Path2)\cdot C(Path2)=0.3\times 24+0.7\times 39=34.5$$

In our point of view, the path complexity based on cognitive informatics is more reasonable than the basic form. From the perspective of system execution, the logic judgment node and service invoking node perhaps will consume more computing time than the common nodes for data transferring or atomic operation, so these kinds of nodes should be assigned to high weights. Form the perspective of process comprehension, the nodes with complicated logic or external service invocation are much harder to be understood than the common node in the process model. Therefore, $AEPC_{CI}$ can precisely

describe the dynamic execution complexity or comprehension difficulty of the Petri net-based process used in Web service composition.

In the above complexity weights of key structure nodes, two cases (i.e., *n*-way branch or parallel structure, $n > 2$) should be considered deeply. Here, take the *n*-way <flow> structure for example, its weight is assigned to 4 in this section. However, considering two-way flow and *n*-way ($n > 2$) flow, cognitively the latter would have higher weight than the former. Thus for *n*-way flow, *n* could also be one factor in deciding the weight of <flow> node.

For the *n*-way <flow> structure, its weight can be refined in the following formula.

$$w(N_{\langle flow \rangle}) = 4 + log_2(n-1), \quad (n \geq 2) \tag{10}$$

Where $N_{\langle flow \rangle}$ represents the parallel structure node, and *n* is the out-degree or in-degree of parallel execution node. In the above formula, we use a logarithm function to describe the cognitive difficulty for the massive parallel execution relations. While considering the branch structure, its weight also can be derived in the similar way.

V. CASE STUDIES

In this section, we will analyze two real-world Web service compositions to validate the feasibility and effectiveness of our measurement methods. The first case named `Online Shop` [17] is a typical composite service, whose business process is described by Figure 4.

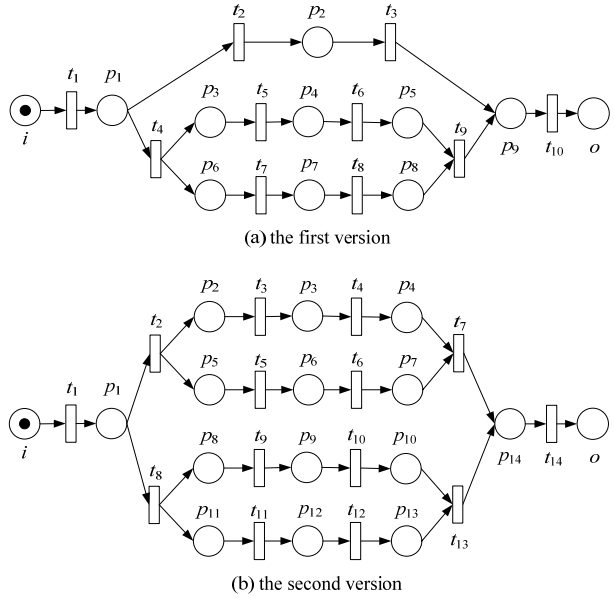

Figure 4. Two versions of business processes for the example application Online Shop

The figure provides two continuous versions of its business process. In the first version, it receives the login message from customer and identifies his/her type. If the message is from an old customer, the process receives the order, and sends an invoicing request to invoicing service. If the message is from a new customer, the process initiates two tasks concurrently. In the first task the process receives the order and then confirms it. In the second task, the process receives the terms of payment before it sends invoicing request to the invoicing service. In the Petri net-based business process, t_3, t_6 and t_8 are the external service nodes. In the second version, the process of old customer is divided into two concurrent tasks. In the first task the process receives the order and then confirms it. In the second task, the process receives which gift is chosen before it sends request to invoicing service. In this new version, t_4, t_6, t_{10} and t_{12} are the external service nodes.

In this case, we suppose the probabilities for old customer and new customer are the same and equal to 0.5. Thus, the metric value of such measurement issues can be calculated and listed in Table 3.

TABLE III. VALUE OF COMPLEXITY METRICS FOR ONLINE SHOP

No.	Metrics	Metric Value	
		1st Version	2nd Version
1	N_P	11	16
2	N_T	10	14
3	N_S	3	4
4	*ADP*	2	2
5	*ADT*	2.2	2.29
6	*TNS*	7.33	8
7	*CC*	3	4
8	*AEPC*	13.5	18
9	$AEPC_{CI}$	20	28

From the results in the above table we can find that, most metrics (except for *ADP*) of the second version are greater than those of the first version. In fact, the business process in Figure 4(b) is the evolved version, so it has more complex control logic than the earlier version. The metric value calculated by our complexity measurement methods can obviously reflects this evolvement feature. That is, our metrics can distinguish the business processes with different complex degree easily. For the complex item of average degree of place (*ADP*), it mainly shows the complexity of data transferring or branch judgment. However, for the business processes in Figure 4(a) and 4(b), they have the same complex degree of branch judgment. Therefore, both of them have the same *ADP* value for two different versions of Online Shop application.

As shown in Figure 5, the second example application is a composite service for travel booking (Here called `Travel Booking`) [18]. At first, it receives the message from customer (i.e., t_1) and checks customer's status by invoking an external service (t_2). Then, it can synchronously call three external services denoted via t_4, t_6 and t_7 for booking car, hotel and flight. Otherwise, it performs the follow-up treatment for the unregistered users. Finally, the booking will be successful with probability of 0.8. Accordingly, the failed probability is 0.2.

For this application, the three basic metrics, i.e., N_P, N_T and N_S, are 14, 14 and 4 respectively. In addition, other metrics can be calculated as follows.

$$ADP(Travel) = \frac{\sum_i deg(p_i)}{N_P} = \frac{31}{14} = 2.21,$$

$$ADT(Travel) = \frac{\sum_i deg(t_i)}{N_T} = \frac{33}{14} = 2.36,$$

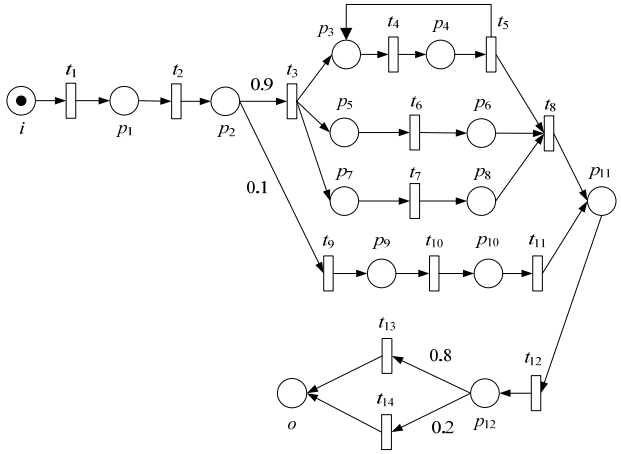

Figure 5. The business process for the composite service of travel booking

$$TNS(Travel) = \frac{|F|}{N_S} = \frac{32}{4} = 8,$$

$$CC(Travel) = 32 - 14 - 14 + 2 = 6.$$

While considering the paths in application `Travel Booking`, there is a loop body between p_3 and t_5. The execution of this loop body can be analyzed in two cases: one time and more than one time. Accordingly, two sub-paths exist from place p_3 to transition t_8: $p_3 \to t_4 \to p_4 \to t_5 \to t_8$ and $p_3 \to t_4 \to p_4 \to t_5 \to p_3 \to t_4 \to p_4 \to t_5 \to t_8$. Here, we use the 2-time loop to represent the case of more than one time execution.

On the other hand, the execution profile of each branch condition is denoted as a label on the corresponding edge, that is,

$prob(p_2 \to t_3) = 0.9$, $prob(p_2 \to t_9) = 0.1$,

$prob(p_{12} \to t_{13}) = 0.8$, $prob(p_{12} \to t_{14}) = 0.8$.

Furthermore, we suppose $prob(t_5 \to p_3) = prob(t_5 \to t_8) = 0.5$ for the loop control condition.

There are six possible paths in the Petri net-based business process, so we can yield the final execution path-based metrics as below.

$$APEC(Travel) = 22 \times 0.9 \times 0.5 \times 0.8 + 22 \times 0.9 \times 0.5 \times 0.2 + 26 \times 0.9 \times 0.5 \times 0.8 + 26 \times 0.9 \times 0.5 \times 0.2 + 15 \times 0.1 \times 0.8 + 15 \times 0.1 \times 0.2 = 23.1,$$

$$APEC_{CI}(Travel) = 38 \times 0.9 \times 0.5 \times 0.8 + 38 \times 0.9 \times 0.5 \times 0.2 + 45 \times 0.9 \times 0.5 \times 0.8 + 45 \times 0.9 \times 0.5 \times 0.2 + 20 \times 0.1 \times 0.8 + 20 \times 0.1 \times 0.2 = 39.35.$$

While comparing the metric results of `Online Shop` with those of `Travel Booking`, we can find that all metric values of the latter application are higher than those of the former. This means that our metrics can reflect the actual situation, because the business process of application `Travel Booking` is more complex than that of `Online Shop`. Therefore, we can claim that our measurement methods are effective and rational.

VI. Related Work

In recent years, complexity analysis work for Web services or their composition has received a lot of attention and there are a number of discussions dedicated to this field. In reference [13,19], J. Cardoso and V. Gruhn et al. have surveyed several contributions for measuring business process models. Here, we only briefly address the existing methods which have closed relations with our work.

At present, BPEL, BPMN and Petri-Net are three main-stream methods to describe the business workflow in WSC. J. Cardoso designed a process complexity metric named control-flow complexity (CFC) to analyze tri-logic workflow [20] through borrowing some ideas from McCabe's cyclomatic complexity. Then, for the BPEL-based process code, he extended his previous work and developed several metrics to characterize some specific perspectives of business process in WSCs. For example, he analyzed the special nodes in BPEL code and assigned different weights to these logic constructs. Compared with their metrics, our work mainly concerns on the complexity analysis for workflow represented by Petri-Net.

BPMN is a widely-adopted denotation to visualize the business process in Web service composition. Accordingly, to analyze the complexity of business workflow expressed by BPMN has cause researcher's attention. Typically, E. Rolón et al. argued several metrics for business process modeled in BPMN [21]. Their metrics are an adaptation and extension of the framework for the modeling and evaluation of software processes (FMESP). In addition, Reijers and Vanderfeesten introduced a heuristic rule to control the proper size of individual activities in business process, and defined a process cohesion and a process coupling metric [22]. Different from their works, we only consider the control flow complexity (i.e., count-based complexity and execution path-based complexity) for Petri net-based process in Web service composition. At present, we have not considered the aspects about cohesion and coupling.

Petri net-based business process representation is firstly proposed by R. Hamadi and B. Benatallah [7]. Compared with BPMN, Petri net is more concise and can express complex parallel execution behaviors. However, parallel is a significant feature of Web service-based system. So using Petri net to describe the composite and dynamic behaviors is very suitable. For example, Zhong et al. used stochastic Petri nets as a solution to the problems of predicting the reliability of web service composition [23]. To the best of our knowledge, our work is the first attempt to assess the complexity of Petri net-based business process in WSC.

From the perspective of program maintainers, complexity can be defined as "difficulty to understand a program or model". Therefore, cognitive informatics can be adopted to understand and measure the fundamental characteristics of program. Using results from cognitive

sciences, Cant et al. come up with a set of tentative complexity metrics for software programs [15]. Wang and Shao [16] defined the cognitive weight as a metric to measure the effort required for comprehending a piece of software. Based on empirical studies, they defined cognitive weights for basic control structures. But, their works are both for the traditional programs, and Web service composition has some differences from them. In this paper, we adopt the cognitive weights for some typical constructs to analyze the execution complexity of some specific path in WSC.

VII. CONCLUSIONS

Web service is a new technology to build distributed applications over the Internet. Although it can greatly increase the reusability of service unit and reduce the coupling between software modules. However, how to analyze and understand Web service-based system with characters such as heterogeneity, distributed and loose-coupling is a difficult task for software maintainers. Therefore, it is very necessary to exploit some precise and reasonable metrics for such system.

In the paper, we adopt Petri net as a graphic notation with formal semantics to describe the control dependence relations between Web services in WSC. Based on this business process representation, two metric set for measuring the control structure complexity of Web service composition is proposed. The fist one is based on the basic count of elements in Petri net-based business process, such as place number, external service number, transfer number per service and so on. The merit of this metric set lies in its simpleness and practicability. The second is execution path-based metric set. We firstly presented the concept of parallel execution relation, and then the execution paths in WSC's process can be deduced. Combined with the execution profile information, path-based complexity metric is addressed. Then, an extension based on cognitive informatics is also discussed. Furthermore, two real applied WSCs are used to confirm the effectiveness and practical value of our metric sets.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 60803046, and the Science Foundation of Jiangxi Educational Committee under Grant No. JJ10433. The paper is an extension of the early short version [24] in proceedings of SOSE 2010.

REFERENCES

[1] W. T. Tsai, X. Bai, and Y. Chen. "On Service-Oriented Software Engineering," Tsinghua University Press, 2008, pp.1-11. (in Chinese)

[2] IBM. "Web Services: Taking e-Business to the Next Level," White Paper, 2000, available from: http://www.ibm.com/developerworks/cn/websphere/download/pdf/e-businessj.pdf

[3] W3C Web Services Activity, available from: http://www.w3.org/2002/ws/, accessed on October 2009.

[4] M. Aoyama, S. Weerawarana, H. Maruyama and et al. "Web Services Engineering: Promises and Chanllenges," Proc. of ICSE'02, Orlando, Florida, USA, ACM Press, 19-25 May 2002, pp. 647-648.

[5] C. Petri. "Kommunikation mit Automaten," PhD Thesis, University of Bonn, Germany, 1962.

[6] J. Peterson. "Petri Net Theory and the Modeling of Systems," Prentice Hall, Englewood Cliffs, 1981.

[7] R. Hamadi and B. Benatallah. "A Petri Net-based Model for Web Service Composition," Proc. of the 14th Australasian Database Conference (ADC'03), Adelaide, Australia, 2003, pp.191-200.

[8] S. Narayanan and S. McIlraith. "Analysis and Simulation of Web Services," Computer Networks, 2003, Vol. 42, pp. 675-693.

[9] OASIS WSBPEL Technical Committee. Web Services Business Process Execution Language, Version 2.0, available at http://docs.oasis-open.org/wsbpel/2.0/wsbpel v2.0.pdf

[10] S. A. White. "Introduction to BPMN," March, 2004. Available at http://www.bpmn.org/Documents/Introduction to BPMN.pdf

[11] Z. Yan and Q. Ding. "Web Service Flow Model ing Based on Petri Net," Computer Applications, 2003, Vol. 23, No. 12, pp. 55-57. (in Chinese)

[12] T. J. McCabe. "A Complexity Measure," IEEE Trans. on Software Engineering, 1976, Vol. 2, No. 4, pp. 308-320.

[13] J. Cardoso, J. Mendling, G. Neumann, and H.A. Reijers. "A Discourse on Complexity of Process Models," Proc. of the 4th Int'l Conference on Business Process Management (BPM'06) Workshops, LNCS 4103, 2006, pp. 117-128.

[14] S. Misra. "A Complexity Measure Based on Cognitive Weights," International Journal of Theoretical and Applied Computer Sciences, Vol. 1, No. 1, 2006, pp. 1-10.

[15] S. N. Cant, D. R. Jeffery and B. Henderson-Sellers. "A Conceptual Model of Cognitive Complexity of Elements of the Programming Process," Information and Software Technology, 1995, Vol.37, No.7, pp. 351-362.

[16] Y. Wang and J. Shao. "A New Measure of Software Complexity Based on Cognitive Weights," Can. J. Elect. Comput. Eng., 2003, Vol.28, No.2, pp.69-74.

[17] D. Wang, B. Li and J. Cai. "Regression Testing of Composite Service: An XBFG-based Approach," Proc. of 2008 IEEE Congress on Services Part II, 6-11 July, 2008, pp. 112-119.

[18] S. A. White. Using BPMN to Model a BPEL Process, 2005, pp.1-18.

[19] V. Gruhn and R. Laue. "Complexity Metrics for Business Process Models," Proc of BIS'06, Klagenfurt, Austria, May 31-June 2, Lecture Notes in Informatics (LNI) 85, GI 2006, 2006, pp. 1-12.

[20] J. Cardoso. "Control-flow Complexity Measurement of Processes and Weyuker's Properties," Transactions on Enformatika, Systems Sciences and Engineering, 2005, Vol. 8, pp. 213-218.

[21] E. Rolón, F. Ruiz, F. García and M. Piattini. "Applying Software Metrics to Evaluate Business Process Models," CLEI Electronic Journal, 2006, Vol.9, No. 1, pp. 1-15.

[22] H. A. Reijers and I. T. P. Vanderfeesten. "Cohesion and Coupling Metrics for Workflow Process Design," Proc. of BPM'04, LNCS 3080, 2004, pp. 290-305.

[23] D. Zhong, Z. Qi, and X. Xu. "Reliability Prediction and Sensitivity Analysis of Web Services Composition," Petri Net: Theory and Applications, 2008, pp. 459-470.

[24] C. Mao. "Complexity Analysis for Petri Net-based Business Process in Web Service Composition," Proc. of the 5th IEEE International Symposium on Service-Oriented System Engineering (SOSE'10), Nanjing, China, 4-5 June, 2010, 4 pages.