future internet

MDPI

Article

A Reliability Calculation Method for Web Service Composition Using Fuzzy Reasoning Colored Petri Nets and Its Application on Supercomputing Cloud Platform

Ziyun Deng [1,2], Lei Chen [2,3,*], Tingqing He [2,4] and Tao Meng [2,4]

[1] Changsha Commerce & Tourism College, Changsha 410116, China; B12090032@hnu.edu.cn
[2] National Supercomputing Center in Changsha, Changsha 410082, China; hetingqin@hnu.edu.cn (T.H.); mengtao@hnu.edu.cn (T.M.)
[3] College of Electrical and Information Engineering, Hunan University, Changsha 410082, China
[4] College of Information Science and Engineering, Hunan University, Changsha 410082, China
* Correspondence: chenleixyz123@hnu.edu.cn; Tel.: +86-158-7495-0836

Academic Editors: Carmen de Pablos Heredero and Dino Giuli
Received: 3 July 2016; Accepted: 20 September 2016; Published: 27 September 2016

Abstract: In order to develop a Supercomputing Cloud Platform (SCP) prototype system using Service-Oriented Architecture (SOA) and Petri nets, we researched some technologies for Web service composition. Specifically, in this paper, we propose a reliability calculation method for Web service compositions, which uses Fuzzy Reasoning Colored Petri Net (FRCPN) to verify the Web service compositions. We put forward a definition of semantic threshold similarity for Web services and a formal definition of FRCPN. We analyzed five kinds of production rules in FRCPN, and applied our method to the SCP prototype. We obtained the reliability value of the end Web service as an indicator of the overall reliability of the FRCPN. The method can test the activity of FRCPN. Experimental results show that the reliability of the Web service composition has a correlation with the number of Web services and the range of reliability transition values.

Keywords: Fuzzy Reasoning Colored Petri Nets (FRCPN); Web service compositions; reliability calculation; Supercomputing Cloud Platform (SCP); Sequential Linked List for Filling Reliability Value (SLLFRV)

1. Introduction

The Supercomputing Cloud Platform (SCP) integrates many different kinds of software including ANSYS, ABAQUS, LS-DYNA, SLURM, LSF, etc. SCP provides a simple Web interactive way for large-scale and parallel CAE simulation tasks. Our research team develops a SCP prototype of supercomputing. Specifically, we use Service-Oriented Architecture (SOA) architecture to develop the prototype. We use Petri nets to describe Web service compositions, and realized the prototype using a workflow system.

For automatic composition for Web services based on semantic technology, we introduce Ontology Web language for Services (OWL-S) to describe Web services. In order to verify the Web Service Compositions, we proposed a reliability calculation method for Web Service Composition Using Fuzzy Reasoning Colored Petri Nets (FRCPN), then we judged whether the Web service composition could meet the reliability requirements.

At present, the applications of OWL-S are well established in the engineering field. The study on web service composition has many theoretical researches and engineering practices, and the study on CPN, FRCPN has also been making progress. However, the reliability of Web service composition needs to be strengthened and studied further.

2. Related Work

Many researchers use SOA architecture, semantic technology, and workflow technology to build Colored Petri Nets (CPN) or composition system for Web services. In [1], the authors developed a platform prototype for cloud simulation named COSIM-CSP using some Web services, discovery technology, and so on. In [2–8], the authors also proposed models for cloud platform and designed system prototypes. In [9], the authors developed a composition system for Web services named OntoPipeliner based on semantics. At present, there are some studies underway on a cloud platform based on OWL-S semantic technology, such as [9–11].

It is one of the main possible directions for studying on Web service compositions base on Petri. Many researchers are dedicated to applying the improvement of Petri nets to combined Web services [12–17]. In [18], the authors proposed a CPN model. In the model, the authors added the transaction features of Web services, considered Quality of Service (QoS) and transaction, and then formed a credible composition for Web services.

In [19], the authors used CPN to design a deterministic analysis method that is used to measure the reliability and the performance. CPN can display the routing structure and apply probability theory. In addition, the authors developed a software tool called WSET, which is used for supporting QoS sensor modeling. In [20], the authors used Petri nets for orchestration with the OWL-S service, which is treated as an ontology engine to support reasoning. In [21], the authors used Hierarchical Colored Petri Net (HCPN) to formally describe Web service compositions, converted the structure of data flow information described by OWL-S, and used the CPN tool to verify the model. In [22], the authors proposed an improved HCPN namely EHCPN to describe Web service compositions. In [23], the authors proposed a method based on translation to convert the process model of OWL-S services into Petri nets. In [24], the authors proposed a method based on stochastic Petri nets for the performance analysis of OWL-S process, and used non-Markov Stochastic Petri Nets as an intermediate representation.

Some scholars have done research on the reliability of Web service compositions using Petri, CPN, or FRCPN [25–29]. In order to improve the reliability value of Web service compositions, [18] and [30] proposed using CPN to formalize the various properties of Web service compositions. In [16], non-Markov Stochastic Petri Nets are proposed based on a probabilistic method of reliability analysis of OWL-S Web service compositions. The authors introduce the normal completion probability of the process to calculate the reliability. In [31], the authors proposed a method to use Petri nets to build reliable Web service compositions. This method verifies the behavior of service compositions and repairs the design errors at design time or run time.

Kim, Y. et al. [32] proposed a trust model expressed by a graph. In the graph, the weights of the edges denote the weights of trust. According to the evaluated trust value, the model can choose a more suitable Web service or Web service composition. In [33], based on the analysis of the data for past services, the authors used the K-mean clustering algorithm to predict the reliability of atomic Web services. Xu, J. et al. [34] developed a model to select the most appropriate service composition. The model uses stochastic Petri nets for service compositions to create a qualitative model, which performs a quantitative analysis of the failure rate and repair rate. In [35], a method called $\lambda - T$ is proposed to quantify the reliability of a repairable system. The main parameters in the method are failure rate, time, and Petri nets. Many methods generate a reachable tree according to the graph, and then calculate the reliability of the tree according to the various parameters of the reachable tree as the reliability of FRCPN graph [36].

In summary, there has been some progress in terms of the reliability calculation method for Web service compositions. However, there is not yet an exact definition of semantic similarity threshold relations in the literature, nor is there a reliability calculation method for Petri nets based on the relations of semantic threshold similarity. The main contributions of this paper are as follows.

- We gradually define the concepts of reliability calculation. Based on the definition of formalized description for Web services, we define the semantic threshold similarity and semantic threshold equality for Web services. We first give the definition of FRCPN and the definition of FRCPN activity. Then, we propose the conditions of the activity. We analyze the reliability calculation of some elements in five kinds of production rules in FRCPN. According to the requirements of reliability calculation for FRCPN, we redefine the formalized definition of transition and the formalized definition of Web services.
- A reliability calculation method for FRCPN is proposed. Firstly, the method calculates the reliability values of the transitions in FRCPN, and uses an algorithm to generate a Sequential Linked List for Filling Reliability Value (SLLFRV) linked list. According to the SLLFRV linked list, we propose an algorithm to calculate the reliability values of all Web services. Finally, we use the reliability value of the end Web service in FRCPN as the reliability value of FRCPN.
- We apply and analyze the reliability calculation method. The reliability value of a FRCPN graph is calculated as an example. We analyze the execution process of the method. We analyze the implementation effect of the method for different size FRCPNs and obtain some rules. We give the application description for the method in SCP.

3. Related Definition

3.1. Definition of Semantic Threshold Similarity and Definition of Semantic Threshold Equality

Definition 1. *Formal description of Web services [9]. A Web service can be formally described as*

$$WS_i(I_i, O_i, P_i, Q_i),$$

where I_i is the input parameter set of Web service WS_i, O_i is the output parameter set of Web service WS_i, P_i is the precondition set of Web service WS_i, and Q_i is the QoS set of Web service WS_i.

Definition 2. *Semantic threshold similarity for two ontologies and semantic threshold equality for two ontologies. There are two ontology concepts, c_i and c_j. By calculating their semantic similarity, we can obtain the value of $sim(c_i, c_j)$. We set a semantic similarity threshold, $k(0 \leq k \leq 1)$. If $sim(c_i, c_j) \geq k$, then the relation of semantic threshold similarity is built between c_i and c_j, denoted as $c_i \sqsupseteq c_j$. If $c_i \sqsupseteq c_j$, and $c_j \sqsupseteq c_i$, then the relation of semantic threshold equality is built between c_i and c_j, denoted as $c_i \cong c_j$.*

Definition 3. *Semantic threshold similarity for two ontology sets and semantic threshold equality for two ontology sets. There are two concept sets $C_i\{c_i\}$ and $C_j\{c_j\}$. We set a semantic similarity threshold, $k(0 \leq k \leq 1)$. If any concept c_i in C_i exists corresponding c_j in C_j to satisfy $c_i \sqsupseteq c_j$, then the relation of semantic threshold similarity is built between C_i and C_j, denoted as $C_i \sqsupseteq C_j$. If $C_i \sqsupseteq C_j$, and $C_j \sqsupseteq C_i$, then the relation of semantic threshold equality is built between C_i and C_j, denoted as $C_i \cong C_j$.*

Definition 4. *Semantic threshold similarity for Web service preconditions and semantic threshold similarity for Web services QoS. There are two Web services, $WS_i(I_i, O_i, P_i, Q_i)$ and $WS_j(I_j, O_j, P_j, Q_j)$; if any logical expression p_i in P_i exists corresponding p_j in P_j, and p_i is **true**, then p_j is **true**, the relation of semantic threshold similarity is built between P_i and P_j, denoted as $P_i \sqsupseteq P_j$. If any logical expression of QoS q_i in Q_i exists corresponding q_j in Q_j, and q_i is **true**, then q_j is **true**, the relation of semantic threshold similarity is built between Q_i and Q_j, denoted as $Q_i \sqsupseteq Q_j$.*

Definition 5. *Semantic threshold similarity for two Web services and semantic threshold equality for two Web services. There are two Web services, $WS_i(I_i, O_i, P_i, Q_i)$ and $WS_j(I_j, O_j, P_j, Q_j)$. We set a semantic similarity threshold $k(0 \leq k \leq 1)$, if $I_i \sqsupseteq I_j$, $O_i \sqsupseteq O_j$, $P_i \sqsupseteq P_j$ and $Q_i \sqsupseteq Q_j$, then the relation of semantic threshold similarity is built between WS_i and WS_j, denoted as $WS_i \sqsupseteq WS_j$. If $WS_i \sqsupseteq WS_j$ and $WS_j \sqsupseteq WS_i$, then the relation of semantic threshold equality is built between WS_i and WS_j, denoted as $WS_i \cong WS_j$.*

3.2. Definition of FRCPN and Definition of FRCPN Activity

Definition 6. *Formal definition of FRCPN based on semantic similarity for Web service composition [16]:*

$$FRCPN = (P, T, F, \mu, PR, Q).$$

(1) $P = \{P_1, P_2, \cdots, P_n\}$ represents the place set, and each place represents a Web service.
(2) $T = \{T_1, T_2, \cdots, T_m\}$ represents the transition set, and each transition represents the matching relationships among Web services.
(3) F is a dependence relation set, represented as an If-Then form.
(4) μ is a function of T, and the function's value is in the range of [0,1], which is the triggered threshold of the transition. The function is also the semantic similarity threshold for Web services, and we can control the minimum triggered threshold using the function.
(5) PR is the precondition set defined on the P, which represents all preconditions being satisfied before the transitions are triggered.
(6) Q is the QoS set defined on the P; it indicates all QoS of the Web services that need to be satisfied before the transitions are triggered.

We can measure the matching degree using semantic similarity between the interfaces of two Web services, and then establish the dependence relation. There are five kinds of dependence relations in FRCPN. The semantic similarity calculation considers some factors, including I/O interfaces P, Q, etc., so we adopt semantic similarity as reliability. It is necessary that the reliability value of the transition achieve semantic threshold k as a trigger.

Definition 7. *FRCPN activity. In FRCPN, if $t \in T$, and a transition sequence exists, the transition sequence is triggered in sequence, and in semantic threshold $k(0 \leq k \leq 1)$, the reliability value Γ of each transition t is satisfied with $\Gamma \geq k$, then transition t is triggered, we consider that transition t is active in semantic threshold k. According to the dependence relations in FRCPN, we can calculate the reliability values of the Web services and the transitions. We use the reliability value of the end Web service in FRCPN as the reliability value ω of FRCPN. In the reliability threshold $a(0 \leq a \leq 1)$, all of the transitions in FRCPN are active in semantic threshold k, and the reliability value ω of FRCPN satisfies $\omega \geq a$, so FRCPN is activity in semantic threshold k and reliability threshold a.*

3.3. Production Rule of FRCPN

Next, we discuss five production rules of FRCPN. According to the reliability values of Web services, we can verify the FRCPN activity, and obtain better quality FRCPN.

3.3.1. The First Production Rule

If WS_i **Then** WS_j (The semantic similarity threshold k)

The first production rule represents a dependence relation $WS_i \rightarrow WS_j$ (the semantic similarity threshold k). The FRCPN based on semantic similarity for Web service compositions is shown in Figure 1.

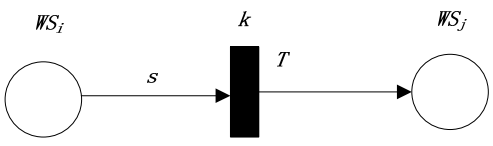

Figure 1. The FRCPN (Fuzzy Reasoning Colored Petri Net) expression of the first production rule.

The above rule indicates the dependence relationship $WS_i \rightarrow WS_j$. Here, s is the semantic similarity value between the outputs of WS_i and the inputs of WS_j. The above rule needs to satisfy some conditions including $s \geq k$, $O_i \supseteq I_j$, and all of the conditions in P_i, Q_i. Therefore,

$$s = m_1 \times \text{sim}(O_i, I_j) + m_2,\ m_1 + m_2 = 1. \tag{1}$$

In Equation (1), $\text{sim}(O_i, I_j)$ represents the semantic similarity between the outputs of WS_i and the inputs of WS_j. m_1 is the weight that represents the attention degree of matching between the outputs of WS_i and the inputs of WS_j. m_2 is the weight that represents the attention degree of satisfying all conditions in P_i and Q_i.

According to the above definition, we define the reliability value Γ_t of the transition T by the semantic similarity between the interfaces of WS_i and the interfaces of WS_j, as follows:

$$\Gamma_t = s. \tag{2}$$

We can define the reliability value of WS_j in this dependence relation as follows:

$$\Gamma_j = \Gamma_i \times \Gamma_t = \Gamma_i \times s. \tag{3}$$

3.3.2. The Second Production Rule

If WS_{i1} **and** WS_{i2} $\cdots$ **and** WS_{in} **Then** WS_j (The semantic similarity threshold k)

The second production rule represents a dependence relation "WS_{i1} **and** WS_{i2} $\cdots$ **and** $WS_{in} \rightarrow WS_j$" (the semantic similarity threshold k). The FRCPN based on semantic similarity for Web service compositions is shown in Figure 2.

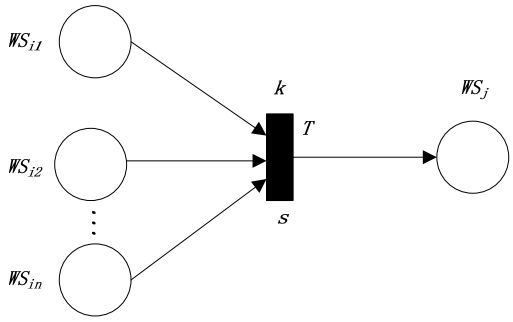

Figure 2. The FRCPN expression of the second production rule.

s is the semantic similarity between $O_{i1} \cup O_{i2} \cup \cdots \cup O_{in}$ (the union of the outputs of "$WS_{i1}, WS_{i2}, \cdots, WS_{in}$") and the inputs of WS_j. The above rule needs to satisfy some conditions, including $s \geq k$, $(O_{i1} \cup O_{i2} \cup \cdots \cup O_{in}) \supseteq I_j$, all of the conditions in "$P_{i1}, P_{i2}, \cdots, P_{in}$", and all of the conditions in "$Q_{i1}, Q_{i2}, \cdots, Q_{in}$". Therefore,

$$s = m_1 \times \text{sim}(O_{i1} \cup O_{i2} \cup \cdots \cup O_{in}, I_j) + m_2,\ m_1 + m_2 = 1. \tag{4}$$

In Equation (4), $\text{sim}(O_{i1} \cup O_{i2} \cup \cdots \cup O_{in}, I_j)$ represents the semantic similarity between $O_{i1} \cup O_{i2} \cup \cdots \cup O_{in}$ and the inputs of WS_j. m_1 is the weight that represents the attention degree of matching between $O_{i1} \cup O_{i2} \cup \cdots \cup O_{in}$ and the inputs of WS_j. m_2 is the weight that represents the attention degree of satisfying all the conditions in "$P_{i1}, P_{i2}, \cdots, P_{in}$" and "$Q_{i1}, Q_{i2}, \cdots, Q_{in}$".

According to the above definition, we define the reliability value Γ_t of the transition T in terms of the semantic similarity between the interfaces of "$WS_{i1}, WS_{i2}, \cdots, WS_{in}$" and the interfaces of WS_j as follows:

$$\Gamma_t = s. \tag{5}$$

We can define the reliability value of WS_j as follows:

$$\Gamma_j = \mathbf{min}\{\Gamma_{i1}, \Gamma_{i2}, \cdots, \Gamma_{in}\} \times \Gamma_t = \mathbf{min}\{\Gamma_{i1}, \Gamma_{i2}, \cdots, \Gamma_{in}\} \times s. \quad (6)$$

3.3.3. The Third Production Rule

If WS_{i1} **or** WS_{i2} $\cdots$ **or** WS_{in} **Then** WS_j (The semantic similarity threshold k)

This production rule represents a dependence relation "WS_{i1} **or** WS_{i2} $\cdots$ **or** $WS_{in} \rightarrow WS_j$" (the semantic similarity threshold k). The FRCPN based on semantic similarity for Web service compositions is shown in Figure 3.

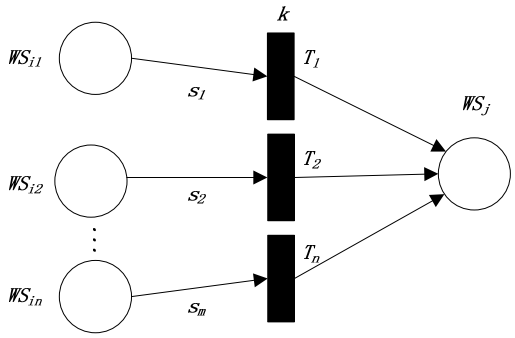

Figure 3. The FRCPN expression of the third production rule.

"s_1, s_2, $\cdots$, s_m" are the values of the semantic similarity between the outputs of "$WS_{i1}, WS_{i2}, \cdots, WS_{in}$," respectively, and the inputs of WS_j. The above rule needs to satisfy some conditions, including $s_1 \geq k, s_2 \geq k, \cdots, s_m \geq k$, $(O_{i1} \sqsupseteq I_j) \vee (O_{i2} \sqsupseteq I_j) \vee \cdots (O_{in} \sqsupseteq I_j)$, all of the conditions in "$P_{i1}, P_{i2}, \cdots, P_{in}$", and those in "$Q_{i1}, Q_{i2}, \cdots, Q_{in}$".

$$\begin{aligned} s_1 &= m_{11} \times \mathrm{sim}(O_{i1}, I_j) + m_{21},\ m_{11} + m_{21} = 1 \\ s_2 &= m_{12} \times \mathrm{sim}(O_{i2}, I_j) + m_{22},\ m_{12} + m_{22} = 1 \\ &\cdots, \\ s_n &= m_{1n} \times sim(O_{in}, I_j) + m_{2n},\ m_{1n} + m_{2n} = 1 \end{aligned} \quad (7)$$

In Equation (7), $\mathrm{sim}(O_{i1}, I_j)$ represents the semantic similarity between the outputs of WS_{i1} and the inputs of WS_j; $\mathrm{sim}(O_{i2}, I_j)$ represents the semantic similarity between the outputs of WS_{i2} and the inputs of WS_j, and so on. m_{11} is the weight that represents the attention degree of matching between the outputs of WS_{i1} and the inputs of WS_j; m_{12} is the weight that represents the attention degree of matching between the outputs of WS_{i2} and the inputs of WS_j, and so on. m_{21} is the weight that represents the attention degree of satisfying all the conditions in P_1 and Q_1; m_{22} is the weight that represents the attention degree of satisfying all the conditions in P_2 and Q_2, and so on.

According to the above definition, we define the reliability values "$\Gamma_1, \Gamma_2, \cdots, \Gamma_n$" of the transitions "$T_1, T_2, \cdots, T_n$" as the semantic similarities between the interfaces of "$WS_{i1}, WS_{i2}, \cdots, WS_{in}$," respectively, and the interfaces of WS_j, as follows:

$$\Gamma_1 = s_1, \Gamma_2 = s_2, \cdots, \Gamma_n = s_n. \quad (8)$$

We can define the reliability of WS_j as follow,

$$\Gamma_j = \mathbf{max}\{\Gamma_{i1} \times \Gamma_1, \Gamma_{i2} \times \Gamma_2, \cdots, \Gamma_{in} \times \Gamma_n\} = \mathbf{max}\{\Gamma_{i1} \times s_1, \Gamma_{i2} \times s_2, \cdots, \Gamma_{in} \times s_n\}. \quad (9)$$

3.3.4. The Fourth Production Rule

If WS_i **then** WS_{j1} **and** WS_{j2} $\cdots$ **and** WS_{jn} (The semantic similarity threshold k)

This fourth production rule represents a dependence relation "$WS_i \rightarrow WS_{j1}$ **and** WS_{j2} $\cdots$ **and** WS_{jn}" (the semantic similarity threshold k). The FRCPN based on semantic similarity for Web service compositions is shown in Figure 4.

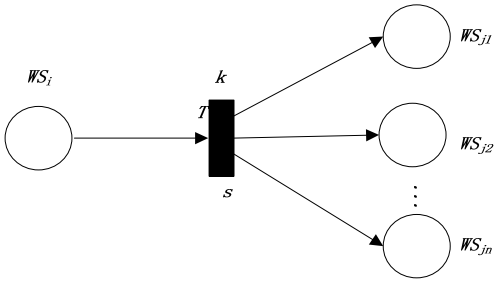

Figure 4. The FRCPN expression of the fourth production rule.

s is the semantic similarity between the outputs of O_i and $I_{j1} \cup I_{j2} \cup \cdots \cup I_{jn}$ (union of inputs of "$WS_{j1}, WS_{j2}, \cdots, WS_{jn}$"). The above rule needs to satisfy some conditions, including $s \geq k$, $O_i \sqsupseteq (I_{j1} \cup I_{j2} \cup \cdots \cup I_{jn})$, and all of the conditions in P_i, Q_i. Therefore,

$$s = m_1 \times \text{sim}(O_i, I_{j1} \cup I_{j2} \cup \cdots \cup I_{jn}) + m_2,\ m_1 + m_2 = 1. \tag{10}$$

In Equation (10), $\text{sim}(O_i, I_{j1} \cup I_{j2} \cup \cdots \cup I_{jn})$ represents the semantic similarity between the outputs of WS_i and $I_{j1} \cup I_{j2} \cup \cdots \cup I_{jn}$. m_1 is the weight that represents the attention degree of matching between the outputs of WS_i and $I_{j1} \cup I_{j2} \cup \cdots \cup I_{jn}$. m_2 is the weight that represents the attention degree of matching meeting all the conditions in P_i and Q_i.

According to the above definition, we define the reliability value Γ_t of the transition T in terms of the semantic similarity between the interfaces of WS_i and the interfaces of $WS_{j1}, WS_{j2}, \cdots, WS_{jn}$, as follows:

$$\Gamma_t = s. \tag{11}$$

We can define the reliability values of "$WS_{j1}, WS_{j2}, \cdots, WS_{jn}$" in this dependence relation as follows:

$$\Gamma_{j1} = \Gamma_{j2} = \cdots = \Gamma_{jn} = \Gamma_i \times \Gamma_t = \Gamma_i \times s. \tag{12}$$

3.3.5. The Fifth Production Rule

If WS_i **then** WS_{j1} **or** WS_{j2} $\cdots$ **or** WS_{jn} (The semantic similarity threshold k)

This fifth production rule represents a dependence relation "$WS_i \rightarrow WS_{j1}$ **or** WS_{j2} $\cdots$ **or** WS_{jn}" (the semantic similarity threshold k). The FRCPN based on semantic similarity for Web service compositions is shown in Figure 5.

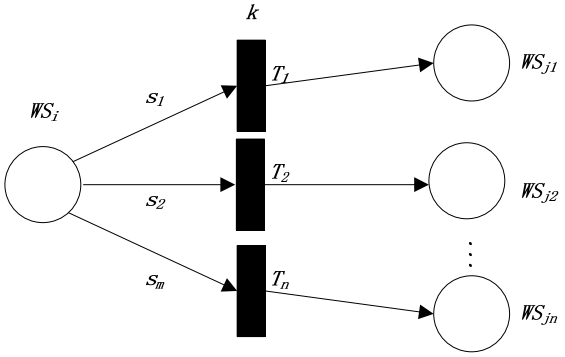

Figure 5. The FRCPN expression of the fifth production rule.

s_1, s_2, $\cdots$, s_n are the semantic similarity between the outputs of WS_i and the inputs of "$WS_{j1}, WS_{j2}, \cdots, WS_{jn}$," respectively. The above rule needs to satisfy some conditions, including $s_1 \geq k, s_2 \geq k, \cdots, s_n \geq k$, $(O_i \sqsupseteq I_{j1}) \vee (O_i \sqsupseteq I_{j2}) \vee \cdots (O_i \sqsupseteq I_{jn})$, and all of the conditions in P_i, Q_i. Therefore,

$$\begin{array}{c} s_1 = m_{11} \times \mathrm{sim}(O_i, I_{j1}) + m_{21},\ m_{11} + m_{21} = 1 \\ s_2 = m_{12} \times \mathrm{sim}(O_i, I_{j2}) + m_{22},\ m_{12} + m_{22} = 1 \\ \cdots \\ s_n = m_{1n} \times \mathrm{sim}(O_i, I_{jn}) + m_{2n},\ m_{1n} + m_{2n} = 1. \end{array} \tag{13}$$

In Equation (13), $\mathrm{sim}(O_i, I_{j1})$ represents the semantic similarity between the outputs of WS_i and the inputs of WS_{j1}; $\mathrm{sim}(O_i, I_{j2})$ represents the semantic similarity between the outputs of WS_i and the inputs of WS_{j2}, and so on. m_{11} is the weight that represents the attention degree of matching between the outputs of WS_i and the inputs of WS_{j1}; m_{12} is the weight that represents the attention degree of matching between the outputs of WS_i and the inputs of WS_{j2}, and so on. m_{21} is the weight that represents the attention degree of satisfying all the conditions in P_i, Q_i in s_1, m_{22} is the weight that represents the attention degree of satisfying all the conditions in P_i, Q_i in s_2, and so on.

According to the above definition, we define the reliability values "$\Gamma_1, \Gamma_2, \cdots, \Gamma_n$" of the transitions "$T_1, T_2, \cdots, T_n$" to equal the semantic similarities between the interfaces of WS_i and the interfaces of "$WS_{j1}, WS_{j2}, \cdots, WS_{jn}$", respectively, as follows:

$$\Gamma_1 = s_1, \Gamma_2 = s_2, \cdots, \Gamma_n = s_n. \tag{14}$$

We can further define the reliability values of $WS_{j1}, WS_{j2}, \cdots, WS_{jn}$. This production rule needs only one Web service WS_i, we can trigger "$WS_{j1}, WS_{j2}, \cdots, WS_{jn}$", so,

$$\begin{array}{c} \Gamma_{j1} = \Gamma_i \times \Gamma_1 = \Gamma_i \times s_1 \\ \Gamma_{j2} = \Gamma_i \times \Gamma_2 = \Gamma_i \times s_2 \\ \cdots \\ \Gamma_{jn} = \Gamma_i \times \Gamma_n = \Gamma_i \times s_n. \end{array} \tag{15}$$

The five kinds of production rule, above have some common points; the points can be summed up with some general rules:

(1) In the first, fourth, and fifth production rules, the reliability value of the postpositional Web service of the transition equals the product of the reliability of the prepositional Web service of the transition and the reliability value of the transition.
(2) In the second production rule, the reliability value of the postpositional Web service of the transition equals the minimum reliability value of all the prepositional Web services multiplied by the reliability value of the transition.
(3) In the third production rule, the reliability value of the postpositional Web service of the transitions equals the maximum value of the reliability values of the respective prepositional Web services multiplied by the reliability values of the transitions.

In order to calculate the reliability of Web services and the reliability values of transitions, we make formal definitions of Web services and transitions in FRCPN.

Definition 8. *Formal definition of Web service in FRCPN.*

$$WS(IRS, PRTS, POTS, PIRS, ReliabilityValue)$$

(1) *WS*, Web Service represents a Web service in FRCPN.
(2) *IRS*, Immediate Reachability Set. For a current Web service WS, if there exists a transition t satisfying some conditions, including $WS \in I(t)$ and $WS_i \in O(t)$, then we call WS_i the immediate

reachability Web service of *WS*. All of the immediate reachability Web services of *WS* are called an immediate reachability set of *WS*, denoted as *IRS*.

(3) *PRTS*, PRepositional Transition Set. For a current Web service *WS*, if there exists a transition t satisfying $WS \in O(t)$, we call t the prepositional transition of *WS*. All of the prepositional transitions of *WS* are called a prepositional transition set of *WS*, denoted as *PRTS*.

(4) *POTS*, POstpositional Transition Set. For a current Web service WS, if there exists a transition t satisfying $WS \in I(t)$, we call t the postpositional transition of *WS*. All of the postpositional transitions of *WS* are called a postpositional transition set of *WS*, denoted as *POTS*.

(5) *PIRS*, Prepositional Immediate Reachability Set. For a current Web service *WS*, if there exists a transition t satisfying some conditions, including $WS \in O(t)$ and $WS_i \in I(t)$, we call WS_i the prepositional immediate reachability Web service of *WS*. All of the prepositional immediate reachability Web services of *WS* are called a prepositional immediate reachability Web service set, denoted as *PIRS*.

(6) *ReliabilityValue*, the reliability value of the current Web service. Before calculating the reliability value of all Web services in the graph, we set the initial Web service's reliability to 1, and the reliability value of other Web services to 0.

Definition 9. *Formal definition of transition in FRCPN.*

$$T(PRWSS, POWSS, Type, Reliability\ Value)$$

(1) *T*, Transition, represents a transition in FRCPN.

(2) *PRWSS*, PRepositional Web Service Set. For a current transition t, if there exists a Web service *WS* satisfying $WS \in I(t)$, we call *WS* the prepositional Web service of t. All of the prepositional Web services of t are called the prepositional Web service set of t, denoted as *PRWSS*.

(3) *POWSS*, POstpositional Web Service Set. For a current transition t, if there exists a Web service *WS* satisfying $WS \in O(t)$, we call *WS* s the postpositional Web Service of t. All of the postpositional Web Services of t are called the postpositional Web Service set of t, denoted as *POWSS*.

(4) *Type*, the type of current transition. There are five types of transition. The first type of transition is shown in Figure 1, the second type of transition is shown in Figure 2, the third type of transition is shown in Figure 3, the fourth type of transition is shown in Figure 4, and the fifth type of transition is shown in Figure 5.

(5) *ReliabilityValue*, the reliability value of the current transition.

4. Reliability Calculation Method for FRCPN

Here we propose a reliability calculation method for Web service compositions in FRCPN, as shown in Figure 6. Firstly, we calculated the reliability values of all the transitions in FRCPN, then we generated the SLLFRV linked list. Finally, according to the reliability values of all the transitions and the SLLFRV linked list, we calculated the reliability value of FRCPN.

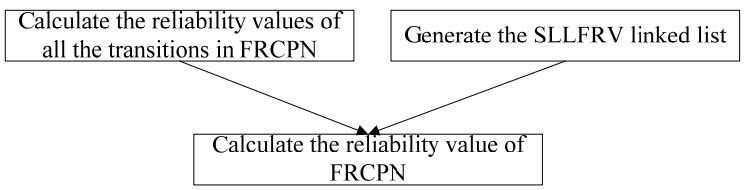

Figure 6. The procedure of the reliability calculation method.

4.1. Reliability Calculation Algorithm for All the Transitions in FRCPN

The time complexity of Algorithm 1 is $O(n)$. After getting the transition set of the graph, the algorithm uses a cycle "**for**" to set the reliability values of the transitions. The reliability values of the transitions are related to the type of transition. If the type of the current transition is the first type, the third type, or the fifth type, and the *PRWSS* of the current transition and the *POWSS* of the current transition have only a Web service, the reliability value of the current transition is $m_1 \times \mathrm{sim}(O_{wsi}, I_{wsj}) + m_2$. If the type of the current transition is the second type, then the *PRWSS* has many Web services, but the *POWSS* of the current transition only has a Web service, so the reliability value of the current transition is $m_1 \times \mathrm{sim}(O_{preWSS1} \cup O_{preWSS2} \cup \cdots \cup O_{preWSSn}, I_{wsj}) + m_2$. If the type of the current transition is the fourth type, then the *POWSS* of the current transition has many Web services, but the *PRWSS* only has a Web service, so the reliability value of the current transition is $m_1 \times \mathrm{sim}(O_{wsi}, I_{postWSS1} \cup I_{postWSS2} \cup \cdots \cup I_{postWSSn}) + m_2$.

Algorithm 1 TSet fillTransitionValue(FRCPN *frcpn*)
Algorithm's name: The reliability calculation algorithm for all the transitions in FRCPN
Input parameter: FRCPN graph *frcpn*
Output parameter: TSet, The transition set filled with reliability values in FRCPN

```
   //Get the transition set of FRCPN graph frcpn
1    TSet tSet = frcpn.getTransitionSet()
   //Set successively the reliability values of all the transitions in frcpn using a loop
2    for int k = 0 to |tSet| − 1 step 1
3        Transition t = tSet.get(k)
         //Get the type of the transition
4        int type = t.getType()
         //Set the reliability value of the transition according to the type of the transition
         //If the type of the transition is the first type, the third type or the fifth type
5        if type = 1 or type = 3 or type = 5 then
             //The PRWSS has only a Web service
6            WS wsi = t.getPRWSS.get(0)
             //The POWSS has only a Web service
7            WS wsj = t.getPOWSS.get(0)
             //Set the reliability value of the transition. In the expression, m1 + m2 = 1
8            t.setReliablityValue(m1 × sim(Owsi, Iwsj) + m2)
9        end if
         //If the type of the transition is the second type
10       if type = 2 then
             //Get the PRWSS of current transition
11           WSSet preWSS = t.getPRWSS()
             //The POWSS has only a Web service
12           WS wsj = t.getPOWSS.get(0)
             //Set the reliability value of the transition. In the expression, m1 + m2 = 1
             //|preWSS| = n
             //The elements in preWSS are represented as preWSS1, preWSS2, ... , preWSSn
13           t.setReliablityValue(m1 × sim(OpreWSS1 ∪ OpreWSS2 ∪ ··· ∪ OpreWSSn, Iwsj) + m2)
14       end if
         // If the type of the transition is the fourth type
15       if type = 4 then
             //The PRWSS has only a Web service

16           WS wsi = t.getPRWSS.get(0)
             //Get the POWSS of current transition
```

Algorithm 1 *Cont*.

```
17          WSSet postWSS = t.getPOWSS ()
            // Set the reliability value of the transition. In the expression, m1 + m2 = 1
            // | postWSS | = n
            //The elements in postWSS are represented as postWSS1, postWSS2, ... , postWSSn
18          t.set Reliablity Value(m1 × sim(Owsi, I_postWSS1 ∪ I_postWSS2 ∪ ··· ∪ I_postWSSn) + m2)
19      end if
20  end for
21  return tSet
```

4.2. Generation Algorithm for SLLFRV Linked List

We set the reliability value of the starting Web service in FRCPN as 1. The call to the Web service is always backward, and a FRCPN is a directed acyclic graph, so we can successively calculate the reliability values of the Web services in FRCPN according to the transitions. When we calculate the last Web service, the reliability value of the last Web service is used as the reliability value of FRCPN. There are the following problems when we set the reliability values of Web services.

(1) If the reliability values of some Web services in the *PIRS* of current Web service are not set, this situation will lead to non-generation of the reliability value for the current Web service.

(2) A transition has several possible postpositional Web Services, and different transitions may have the same postpositional Web Service, so we need to consider the situation for setting the reliability value of Web service many times.

Regarding the first problem, we propose a SLLFRV linked list. According to the sequence of transitions in SLLFRV, we can successively set the reliability values in the *POWSS* of the current transition. Certainly, this solves the first problem, because the reliability values in *PIRS* of the current transition have not been generated.

Regarding the second problem, we propose a reliability calculation algorithm for Web services in FRCPN. In the algorithm, we consider the situation for setting the reliability value of Web service many times. If we arrive at the current Web service from multiple different paths, we select a higher reliability value among multiple paths as the reliability value of the current Web service.

The data structure of SLLFRV is a simple Singly Linked List, denoted as LinkT. The algorithm for generating SLLFRV is given below.

Algorithm 2 uses the recursive idea. Each time, the algorithm performs a double cycle "**for**" and continues recursion, so the time complexity of the algorithm is $O(n^3)$.

Algorithm 2 LinkT generateLinkT(FRCPN *frcpn*, Transition *t*, LinkT *linkT*)
Algorithm's name: The generation algorithm for SLLFRV
Input parameters: FRCPN graph *frcpn*, current transition *t*, SLLFRV linked list *linkT*. When the algorithm is called first, we set the input parameter *t* is ∅, and set the input parameter *linkT* is ∅.
Output parameter: the SLLFRV linked list generated

```
  //If t is ∅ and linkT is ∅, then it is the first time to call this algorithm
1   if t is ∅ and linkT is ∅ then
2       linkT = newLinkT()
        //Get starting Web service of frcpn
3       WS ws = frcpn.getStartWebService()
        //Get the POTS of starting Web service
4       TSet postTSet = ws.getPOTS()
5       for int i = 0 to | postTSet | − 1 step 1
6           if i = 0 then
7               linkT = postTSet.get(0)
8               linkT.top = postTSet.get(0)
9               linkT.last = postTSet.get(0)
10          else
```

Algorithm 2 *Cont*.

```
11          linkT.last.next = postTSet.get(i)
12          linkT.last = postTSet.get(i)
13              end if
14          end for
            //Continue to iterative generate linkT
15          for int i = 0 to |postTSet| − 1 step 1
16              generateLinkT(fcpn,postTSet.get(i),linkT)
17          end for
18      else //It is not first time to call generateLinkT()
            // Generate linkT and return to the superior function
19          if t is ∅ then
20          return linkT
21          end if
            //Get the POWSS of the current transition
22          WSSet wsSet = t.getPOWSS ()
23          for int i = 0 to |wsSet| − 1 step 1
24              WS ws = wsSet.get(i)
                //Get the POTS of the current Web service
25              TSet postTSet = ws.getPOTS()
                //Get the PRTS of the current Web service
26              TSet preTSet = ws.getPRTS()
                //If PRTS of the current Web service is not in linkT
27              if preTSet not in linkT then
28          break
29          end if
                //If the current Web service does not have a postpositional transition
30              if postTSet is ∅ then
31          break
32          end if
33              for int j = 0 to |postTSet| − 1 step 1
34                  Transition ct = postTSet.get(j)
                    //Get the union of POWSS of all the transitions in linkT
35                  WSSet linkTPreWSSet = ∪_{k=0}^{|linkT|} linkT.get(k).getPOWSS()
                    //Set the initial value of the judgment flag to false
                    /*The judgment flag indicates whether the PRWSS of the current transition is
included in preWSSet*/
36                  boolean flag = false
37                  WSSet tPreWSSet = ct.getPRWSS()
                    // If tPreWSSet is a subset of linkTPreWSSet, then set the flag to true
38                  if tPreWSSet in linkTPreWSSet then
39                  flag = true
40                  end if
                    /*If the current transition is not included in linkT, and the judgment flag is
                    true*/
41                  if ct not in linkT and flag is true then
42              linkT.last.next = ct
43              linkT.last = ct
44                  end if
45              end for
                // Continue to iterative generate linkT
46              for int i = 0 to |postTSet| − 1 step 1
47                  generateLinkT(fcpn,postTSet.get(i),linkT)
48          end for
49          end for
50          end if
51          return linkT
```

When the algorithm is first time to be called, t and *linkT* are ∅, so the algorithm creates a new *linkT*, and begins to inspect from the starting Web service of the graph. The algorithm puts the *POTS* of the starting Web service into *linkT*, and then continues the recursion.

If the algorithm is not first time to be called, then the algorithm must judge some things. If the current transition t is ∅, then it indicates that recursion is completed and the algorithm can return to the superior function. If the transition t is not empty, then the algorithm gets the *POWSS* of the current transition and uses a cycle "**for**" to inspect the elements in *POWSS*.

Next, the algorithm gets the POTS *postTSet* of the current Web service and the PRTS *preTSet* of the current Web service. If *preTSet* is not in *linkT*, it shows that the reliability values of the prepositional transitions of the current Web service and the reliability values of the postpositional transitions of the current Web service have not all been generated, so the algorithm does not continue to inspect the candidate transitions, and uses "**break**" to exit inspection. If the current Web service does not have an immediate postpositional transition, *postTSet* is empty and the algorithm does not continue to follow the transition in the future, so also uses "**break**" to exit inspection.

Next, the algorithm uses a cycle "**for**" to traverse the transitions in *postTSet*. In the ideas of the algorithm, if the transitions are already in *linkT*, then the reliability values of all the immediate postpositional web services of the transitions can be calculated. The algorithm first gets the union of *POWSS* of all the transitions in *linkT*, and then executes set operations. If the *PRWSS* of the current transition is a subset of *linkTPreWSSet*, then it indicates that the reliability values of all the immediate postpositional Web services of the current transition can be calculated. If the current transition is not in *linkT*, the algorithm puts the current transition into *linkT* and then continues recursion.

In summary, if the transition is in *linkT*, the reliability values of all the prepositional Web services of the transition can be calculated according to the order in *linkT* before the reliability values of the postpositional Web services of the transition are calculated. Therefore, the algorithm can confirm that the reliability values of the immediate postpositional Web services of the transition can be calculated.

4.3. Reliability Calculation Algorithm for All the Web Services in FRCPN

Algorithm 3 was used to calculate the reliability values of Web Services according to SLLFRV *linkT* in sequence. According to the content of the algorithm, the time complexity of the algorithm is $O(n^2)$.

Algorithm 3 FRCPN fillWSValue(FRCPN *frcpn*, LinkT *linkT*)
Algorithm's name: the reliability calculation algorithm for FRCPN *frcpn*
Input parameters: FRCPN graph *frcpn*, SLLFRV linked list *linkT*
Output parameter: FRCPN graph *frcpn* that the graph has generated the reliability values of Web services

```
   //According to SLLFRV linkT, generate successively the reliability values of Web services
1  for Transition t = linkT.top to t = null step t = t.next
       //If the current transition is of the first, fourth, or fifth type
2      if t.getType() = 1 or t.getType() = 4 or t.getType() = 5 then
           /*Get the reliability value of the prepositional Web service of the current transition*/
3          float trustValue = t.getPRWSS().get(0).getTrustValue() × t.getTrustValue()
           //Get the POWSS of the transition t
4          WSSet postWSSet = t.getPOWSS ()
           //Set the reliability values of the Web services in POWSS of transition t
5          for int i = 0 to |postWSSet| − 1 step 1
6              WS ws = postWSSet.get(i)
7              ws.setTrustValue( Max{ ws.getTrustValue, trustValue} )
8          end for
```

Algorithm 3 *Cont*.

```
9     end if
      //If the current transition is of the second type
10    if t.getType() = 2 then
        //Get the PRWSS of the current transition
11      WSSet preWSSet = t.getPRWSS()
            /*The reliability value of the postpositional Web service of the transition equals the
minimum value of reliability values of all the prepositional Web services of the transition*/
        //|preWSSet| =
12      float trustValue = Min( preWSSet.get(0).getTrustValue(),
          preWSSet.get(1).getTrustValue(), ... ,preWSSet.get(n).getTrustValue())
          ×t.getTrustValue()
        //The POWSS of the current transition has only one Web service
13      WS ws = t.getPOWSS.get(0)
14      ws.setTrustValue( Max{ ws.getTrustValue, trustValue} );
15    end if
      // If the current transition is of the third type
16    if t.getType() = 3 then
        //The POWSS of the current transition only has one Web service
17    WS ws = t.getPOWSS().get(0)
        //Get the PRTS of the current transition
18    TSet tSet = ws.getPRTS()
        //Get the reliability value of ws, |tSet| = n
19    float trustValue = Max(
          tSet.get(0).getPRWSS().get(0).getTrustValue()×tSet.get(0).getTrustVale(),
          tSet.get(1).getPRWSS().get(0).getTrustValue()×tSet.get(1).getTrustVale(),
        ... ,
          tSet.get(n).getPRWSS().get(0).getTrustValue()×tSet.get(n).getTrustVale());
20    ws.setTrustValue( Max{ ws.getTrustValue, trustValue} );
21    end if
22  end for
23  return frcpn
```

The algorithm traverses the transitions in SLLFRV *linkT*, in order to calculate the reliability values of the direct subsequent Web services of the transitions in SLLFRV *linkT* according to the type of transition. If the current transition t is of the first, fourth, or fifth type, it only has one immediate prepositional Web service, and the reliability values of immediate postpositional Web services are t.getPRWSS().get(0).getTrustValue()$\times t$.getTrustValue(). In order to prevent conflict between the reliability values of multiple transitions, the algorithm executes the operation "**Max**", so as to always get the highest reliability value as the final reliability value of a Web service.

If the current transition t is of the second type, the current transition has multiple immediate prepositional Web services, so the algorithm must take the minimum reliability value of the immediate prepositional Web services as the basis for calculating the reliability values of the postpositional Web services of the current transition. The calculation formula is **Min**(*preWSSet*.get(0).getTrustValue(), *preWSSet*.get(1).getTrustValue(), ... ,*preWSSet*.get(n).getTrustValue())$\times t$.getTrustValue().

If the current transition t is of the third type, the current transition has only one immediate postpositional Web service, so the algorithm must take the operation "**Max**" after the reliability values of the immediate prepositional Web Services and the reliability value of the current transition are multiplied. The algorithm can obtain the reliability value of the immediate postpositional Web service. The calculation formula is

$$\textbf{Max}(tSet.\text{get}(0).\text{getPRWSS}().\text{get}(0).\text{getTrustValue}() \times tSet.\text{get}(0).\text{getTrustValue}(),\_$$

$$tSet.\text{get}(1).\text{getPRWSS}().\text{get}(0).\text{getTrustValue}() \times tSet.\text{get}(1).\text{getTrustValue}(), \ldots ,\_$$

$$tSet.\text{get}(n).\text{getPRWSS}().\text{get}(0).\text{getTrustValue}() \times tSet.\text{get}(n).\text{getTrustValue}(),).$$

5. An Example of the Application of the Method

Assume that we have generated a service composition expressed by FRCPN for the calculation process for CAE simulation in SCP prototype, as shown in Figure 7. According to the reliability calculation method, we will calculate the reliability value of the FRCPN.

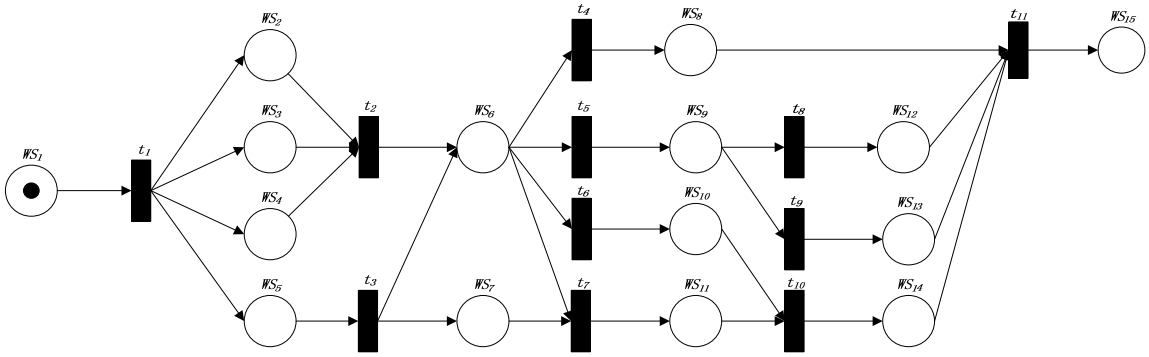

Figure 7. The FRCPN generated.

We get all kinds of parameters of Web services, as shown in Table 1. The reliability value *ReliabilityValue* is required for the calculation, so we use the method in this paper to calculate the reliability value.

Table 1. The parameters of Web services in FRCPN (Fuzzy Reasoning Colored Petri Net).

Web Services' Name	IRS (Immediate Reachability Set)	PRTS (PRepositional Transition Set)	POTS (POstpositional Transition Set)	PIRS (Prepositional Immediate Reachability Set)
WS_1	$\{WS_2, WS_3, WS_4, WS_5\}$	$\varnothing$	$\{t_1\}$	$\varnothing$
WS_2	$\{WS_6\}$	$\{t_1\}$	$\{t_2\}$	$\{WS_1\}$
WS_3	$\{WS_6\}$	$\{t_1\}$	$\{t_2\}$	$\{WS_1\}$
WS_4	$\{WS_6\}$	$\{t_1\}$	$\{t_2\}$	$\{WS_1\}$
WS_5	$\{WS_6, WS_7\}$	$\{t_1\}$	$\{t_3\}$	$\{WS_1\}$
WS_6	$\{WS_4, WS_5, WS_6, WS_7\}$	$\{t_2,t_3\}$	$\{t_4,t_5,t_6,t_7\}$	$\{WS_2,WS_3,WS_4,WS_5\}$
WS_7	$\{WS_{11}\}$	$\{t_3\}$	$\{t_7\}$	$\{WS_5\}$
WS_8	$\{WS_{15}\}$	$\{t_4\}$	$\{t_{11}\}$	$\{WS_6\}$
WS_9	$\{WS_{12}, WS_{13}\}$	$\{t_5\}$	$\{t_8,t_9\}$	$\{WS_6\}$
WS_{10}	$\{WS_{14}\}$	$\{t_6\}$	$\{t_{10}\}$	$\{WS_6\}$
WS_{11}	$\{WS_{14}\}$	$\{t_7\}$	$\{t_{10}\}$	$\{WS_6,WS_7\}$
WS_{12}	$\{WS_{15}\}$	$\{t_8\}$	$\{t_{11}\}$	$\{WS_9\}$
WS_{13}	$\{WS_{15}\}$	$\{t_9\}$	$\{t_{11}\}$	$\{WS_9\}$
WS_{14}	$\{WS_{15}\}$	$\{t_{10}\}$	$\{t_{11}\}$	$\{WS_{10},WS_{11}\}$
WS_{15}	$\varnothing$	$\{t_{11}\}$	$\varnothing$	$\{WS_8,WS_{12},WS_{13},WS_{14}\}$

Firstly, we use Algorithm 1 to get the reliability values of the transitions in the FRCPN. We use *frcpn*.getTransitionSet() to get the transition sequence, which is $\{t_4,t_7,t_5,t_6,t_8,t_{11},t_9,t_{10},t_3,t_2,t_1\}$. We set the semantic similarity threshold $k = 0.8$, $m_1 = 0.8$, and $m_2 = 0.2$ in all the formulas. All of the conditions in P, Q are satisfied. All the transitions in the FRCPN are shown in Table 2.

Table 2. The parameters of the transitions in the FRCPN.

Transition's Name	Known Similarity	PRWSS	POWSS	Type
t_1	$sim(O_{WS1}, I_{WS2} \cup I_{WS3} \cup I_{WS4} \cup I_{WS5}) = 0.94$	$\{WS_1\}$	$\{WS_2,WS_3,WS_4,WS_5\}$	fourth
t_2	$sim(O_{WS2} \cup O_{WS3} \cup O_{WS4}, I_{WS6}) = 0.92$	$\{WS_2,WS_3,WS_4\}$	$\{WS_6\}$	second
t_3	$sim(O_{WS5}, I_{WS6} \cup I_{WS7}) = 0.9$	$\{WS_5\}$	$\{WS_6,WS_7\}$	fourth
t_4	$sim(O_{ws6}, I_{ws8}) = 0.98$	$\{WS_6\}$	$\{WS_8\}$	fifth
t_5	$sim(O_{ws6}, I_{ws9}) = 0.88$	$\{WS_6\}$	$\{WS_9\}$	fifth
t_6	$sim(O_{ws6}, I_{ws10}) = 0.92$	$\{WS_6\}$	$\{WS_{10}\}$	fifth
t_7	$sim(O_{WS6} \cup O_{WS7}, I_{ws11}) = 0.9$	$\{WS_6,WS_7\}$	$\{WS_{11}\}$	second
t_8	$sim(O_{ws9}, I_{ws12}) = 0.94$	$\{WS_9\}$	$\{WS_{12}\}$	fifth
t_9	$sim(O_{ws9}, I_{ws13}) = 0.94$	$\{WS_9\}$	$\{WS_{13}\}$	fifth
t_{10}	$sim(O_{WS10} \cup O_{WS11}, I_{ws14}) = 0.9$	$\{WS_{10},WS_{11}\}$	$\{WS_{14}\}$	second
t_{11}	$sim(O_{WS8} \cup O_{WS12} \cup O_{WS13} \cup O_{WS14}, I_{15}) = 0.92$	$\{WS_8,WS_{12},WS_{13},WS_{14}\}$	$\{WS_{15}\}$	second

According to the sequence $\{t_4,t_7,t_5,t_6,t_8,t_{11},t_9,t_{10},t_3,t_2,t_1\}$ and Algorithm 1, we calculate the reliability values of the transitions. Let us start with a look at the calculation of t_4's reliability value. We know from Table 2 that t_4 is of the fifth type. According to Algorithm 1, *wsi* and *wsj* are WS_6 and WS_8, respectively, and we can calculate the reliability value of t_4 as follows:

$$ReliabilityValue(t_4) = m_1 \times \mathrm{sim}(O_{wsi}, I_{wsj}) + m_2 = m_1 \times \mathrm{sim}(O_{WS6}, I_{WS8}) + m_2 = 0.8 \times 0.98 + 0.2 = 0.984.$$

According to Algorithm 1, we can calculate the reliability value of each transition in order as follows:

$$\begin{aligned} ReliabilityValue(t_7) &= m_1 \times \mathrm{sim}(O_{preWSS1} \cup O_{preWSS2} \cup \cdots \cup O_{preWSSn}, \mathrm{I_{wsj}}) + m_2 \\ &= m_1 \times \mathrm{sim}(O_{WS6} \cup O_{WS7}, I_{WS11}) + m_2 = 0.8 \times 0.9 + 0.2 = 0.92 \end{aligned}$$

$$ReliabilityValue(t_5) = m_1 \times \mathrm{sim}(O_{wsi}, I_{wsj}) + m_2 = m_1 \times \mathrm{sim}(O_{WS6}, I_{WS9}) + m_2 = 0.8 \times 0.88 + 0.2 = 0.904$$

$$ReliabilityValue(t_6) = m_1 \times \mathrm{sim}(O_{wsi}, I_{wsj}) + m_2 = m_1 \times \mathrm{sim}(O_{WS6}, I_{WS10}) + m_2 = 0.8 \times 0.92 + 0.2 = 0.936$$

$$ReliabilityValue(t_8) = m_1 \times \mathrm{sim}(O_{wsi}, I_{wsj}) + m_2 = m_1 \times \mathrm{sim}(O_{WS9}, I_{WS12}) + m_2 = 0.8 \times 0.94 + 0.2 = 0.952$$

$$\begin{aligned} ReliabilityValue(t_{11}) &= m_1 \times \mathrm{sim}(O_{preWSS1} \cup O_{preWSS2} \cup \cdots \cup O_{preWSSn}, I_{wsj}) + m_2 \\ &= m_1 \times \mathrm{sim}(O_{WS8} \cup O_{WS12} \cup O_{WS13} \cup O_{WS14}, I_{15}) + m_2 = 0.8 \times 0.92 + 0.2 = 0.936 \end{aligned}$$

$$ReliabilityValue(t_9) = m_1 \times \mathrm{sim}(O_{wsi}, I_{wsj}) + m_2 = m_1 \times \mathrm{sim}(O_{WS9}, I_{WS13}) + m_2 = 0.8 \times 0.94 + 0.2 = 0.952$$

$$\begin{aligned} ReliabilityValue(t_{10}) &= m_1 \times \mathrm{sim}(O_{preWSS1} \cup O_{preWSS2} \cup \cdots \cup O_{preWSSn}, I_{wsj}) + m_2 \\ &= m_1 \times \mathrm{sim}(O_{WS10} \cup O_{WS11}, I_{WS14}) + m_2 = 0.8 \times 0.9 + 0.2 = 0.92 \end{aligned}$$

$$\begin{aligned} ReliabilityValue(t_3) &= m_1 \times \mathrm{sim}(O_{wsi}, I_{postWSS1} \cup I_{postWSS2} \cup \cdots \cup I_{postWSSn}) + m_2 \\ &= m_1 \times \mathrm{sim}(O_{WS5}, I_{WS6} \cup I_{WS7}) + m_2 = 0.8 \times 0.9 + 0.2 = 0.92 \end{aligned}$$

$$\begin{aligned} ReliabilityValue(t_2) &= m_1 \times \mathrm{sim}(O_{preWSS1} \cup O_{preWSS2} \cup \cdots \cup O_{preWSSn}, I_{wsj}) + m_2 \\ &= m_1 \times \mathrm{sim}(O_{WS2} \cup O_{WS3} \cup O_{WS4}, I_{WS6}) + m_2 = 0.8 \times 0.92 + 0.2 = 0.936 \end{aligned}$$

$$\begin{aligned} ReliabilityValue(t_1) &= m_1 \times \mathrm{sim}(O_{wsi}, I_{postWSS1} \cup I_{postWSS2} \cup \cdots \cup I_{postWSSn}) + m_2 \\ &= m_1 \times \mathrm{sim}(O_{WS1}, I_{WS2} \cup I_{WS3} \cup I_{WS4} \cup I_{WS5}) + m_2 = 0.8 \times 0.94 + 0.2 = 0.952 \end{aligned}$$

We continue to use Algorithm 2 to generate the SLLFRV linked list. When the algorithm is first time to be called, we can use generateLinkT(*frcpn*,$\varnothing$,$\varnothing$). Therefore, we get $postTSet = \{t_1\}$, and the algorithm puts t_1 in the to SLLFRV linked list *linkT*. Next, the algorithm calls recursively generateLinkT(*frcpn*,t_1,*linkT*).

When the algorithm is called by generateLinkT(*frcpn*,t_1,*linkT*), we get $wsSet = \{WS_2, WS_3, WS_4, WS_5\}$. Next, according to the cycle "**for**" to *wsSet* in the algorithm, we get

$postTSet = \{t_2\}$, $preTSet = \{t_1\}$, and know that the value of "*pretest* **not in** *linkT*" is **false**. In the cycle "**for**" to *postTSet* in the algorithm, we can get $ct = t_2$ and the following results:

$$linkTPreWSSet = \bigcup_{k=0}^{|linkT|} linkT.\text{get}(k).\text{getPOWSS}() = \{WS_2, WS_3, WS_4, WS_5\}$$

$$tPreWSSet = ct.\text{getPRWSS}(\ ) = \{WS_2, WS_3, WS_4\}.$$

According to the above analysis, *tPreWSSet* is a subset of *linkTPreWSSet*, the value of flag is **true**. "*ct* **not in** *linkT*" indicates that the current transition is not in *linkT*, and the value of flag is **true**, so the algorithm puts t_2 into the tail of *linkT*.

Next, the algorithm is called recursively generateLinkT(*frcpn*,t_2,*linkT*). Through continuous recursive calls, the algorithm produces the SLLFRV *linkT* as follows:

$$t_1 \to t_2 \to t_3 \to t_4 \to t_5 \to t_7 \to t_8 \to t_9 \to t_{10} \to t_{11}$$

According to the sequence in the SLLFRV *linkT*, we use Algorithm 3 to get the reliability values of all the Web services as follows:

$$TrustValue(WS_1) = 1.0$$

$$Type(t_1) = 4,\ TrustValue(WS_2) = \mathbf{Max}(1.0 \times 0.952, 0) = 0.952,$$

$$Type(t_1) = 4,\ TrustValue(WS_2) = \mathbf{Max}(1.0 \times 0.952, 0) = 0.952,$$

$$TrustValue(WS_3) = \mathbf{Max}(1.0 \times 0.952, 0) = 0.952,$$

$$TrustValue(WS_4) = \mathbf{Max}(1.0 \times 0.952, 0) = 0.952$$

$$TrustValue(WS_5) = \mathbf{Max}(1.0 \times 0.952, 0) = 0.952$$

$$Type(t_2) = 2,\ TrustValue(WS_6) = \mathbf{Max}(\text{Min}(0.952, 0.952, 0.952) \times 0.936, 0) = 0.891$$

$$Type(t_3) = 4, TrustValue(WS_6) = \mathbf{Max}(0.952 \times 0.92, 0.891) = 0.891$$

$$TrustValue(WS_7) = \mathbf{Max}(0.952 \times 0.92, 0) = 0.876$$

$$Type(t_4) = 5,\ TrustValue(WS_8) = \mathbf{Max}(0.891 \times 0.984, 0) = 0.877$$

$$Type(t_5) = 5,\ TrustValue(WS_9) = \mathbf{Max}(0.891 \times 0.904, 0) = 0.805$$

$$Type(t_6) = 5,\ TrustValue(WS_{10}) = \mathbf{Max}(0.891 \times 0.936, 0) = 0.834$$

$$Type(t_7) = 2, TrustValue(WS_{11}) = \mathbf{Max}(\text{Min}(0.891, 0.876) \times 0.92, 0) = 0.806$$

$$Type(t_8) = 5,\ TrustValue(WS_{12}) = \mathbf{Max}(0.805 \times 0.952, 0) = 0.766$$

$$Type(t_9) = 5,\ TrustValue(WS_{13}) = \mathbf{Max}(0.805 \times 0.952, 0) = 0.766$$

$$Type(t_{10}) = 2,\ TrustValue(WS_{14}) = \mathbf{Max}(\mathbf{Min}(0.834, 0.806) \times 0.92, 0) = 0.791$$

$$Type(t_{11}) = 2,\ TrustValue(WS_{15}) = \mathbf{Max}(\mathbf{Min}(0.877, 0.766, 0.766, 0.791) \times 0.936, 0) = 0.717.$$

The last Web service is WS_{15} in the FRCPN graph, so the reliability of the FRCPN graph is the reliability value of WS_{15}. The reliability value of the FRCPN graph is 0.717.

6. Simulated Experiments

In order to analyze the effect of the method proposed in this paper, we used the SCP prototype to conduct simulated experiments. Due to the uncertainty of the Web services, under the premise of the validity of experiment, we made the following settings for the experimental conditions and related parameters.

We used seven FRCPN graphs to simulate the Web service compositions; the number of Web services in the FRCPN graphs is 10, 30, 50, 70, 90, 110, and 130, respectively. We used five kinds of production rules to represent FRCPN, but kept the ratio between Web services and transitions is 1:5, so the length of SLLFRV is respectively 2, 6, 10, 14, 18, 22. The similarity threshold of all the transitions is set to the same value. After Algorithm 1 is finished, we assumed that the reliability values of transitions are divided into three ranges, i.e., $(0.95, 1.0]$, $(0.90, 0.95]$, and $(0.85, 0.90]$. According to the three ranges, we can get three sets of data; the similarity thresholds of the three sets are 0.95, 0.90, and 0.85, respectively.

The purpose of the experiment was to simulate the reliability values of FRCPN in different reliability ranges of transitions. The reliability values of Web service compositions were unlimited. The experimental results are shown in Figure 8. In the figure, the horizontal axis represents the number of Web services in FRCPN; the vertical axis represents the reliability values of Web service compositions.

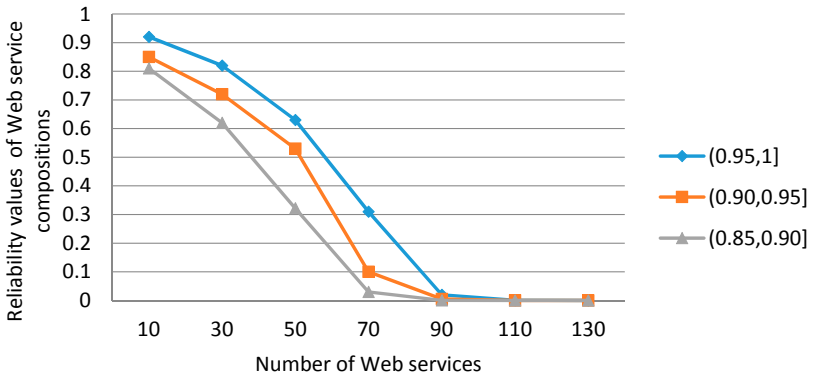

Figure 8. The results of the simulated experiment.

According to Figure 8, we can know the relationships among the reliability values of Web service compositions, the number of Web services, and the reliability ranges of transitions. The following relationships can be obtained:

(1) Under the same reliability range of transitions, if the number of Web services is greater, then the length of SLLFRV is longer and the reliability values of the Web service compositions are smaller.

(2) Under different reliability ranges of transitions, if the minimum range is smaller, then as the number of Web services increases, the reliability values of the service compositions decrease more rapidly.

(3) We should proper control the scale of Web services in FRCPN. For example, if we set the reliability threshold of Web service compositions as 0.6, then in the reliability range (0.95, 1.0] of transitions, the number of Web services should be controlled below 30 and the length of SLLFRV is below 6.

7. Application in SCP

The SCP prototype uses Eclipse as a development tool, Java as a development language, OWL-S as a Web service description language, Tuscany as the Web services' container, WSDL2OWLS as conversion tool, and OWL-S API as an interface tool.

The Web services in the SCP prototype are implemented by POJO Java class, and we use Web Service Description Language (WSDL) to describe the Web services. The software of the SCP prototype generates semantic Web service descriptions after filling QoS content according to WSDL, and then gets OWL documents.

The users of the SCP prototype submit their computing requirements for CAE simulation through the Web system, as shown in Figure 9. After the SCP prototype receives the user's request, the SCP prototype generates the request ontology, and then generates FRCPN. Therefore, we used the method in this paper to calculate the reliability value, then screen and get the best FRCPN.

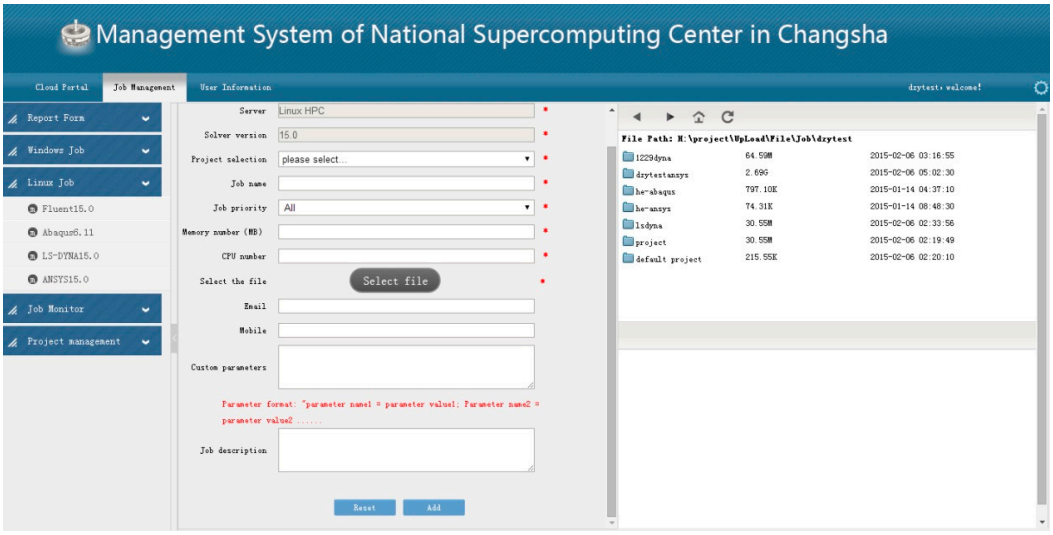

Figure 9. The interface for submitting a job in the SCP (Supercomputing Cloud Platform) prototype.

After the SCP prototype generates FRCPN graphs for CAE simulation, many FRCPN graphs are selected. The SCP prototype then calculates the reliability value of FRCPN graphs. We set $k = 0.95$, $a = 0.9$, and the number of Web services as not more than 30. We chose the highest reliability of FRCPN as the final business flow chart. Then we used Activiti to implement FRCPN. An example Activiti workflow for a computing job is shown in Figure 10.

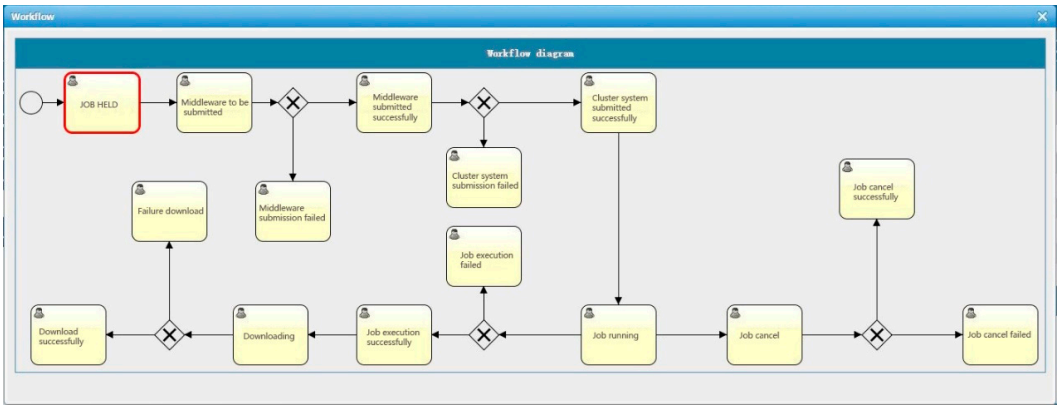

Figure 10. An example Activiti workflow for a computing job in the SCP prototype.

8. Conclusions and Further Work

When we use SOA architecture to develop an SCP prototype, we need to verify FRCPN after the prototype automatically generates FRCPN, and calculate the reliability value of FRCPN for further screening. We proposed a series of definitions, including semantic threshold similarity for Web services, formal definition of transition in FRCPN, etc. We proposed a reliability calculation method for Web service compositions using FRCPN. Firstly, the method calculates the reliability values of all the transitions in FRCPN. Then, the method generates the SLLFRV linked list. Finally, according to the reliability values of all the transitions and SLLFRV linked list, the method calculates the reliability value of FRCPN.

The experiment shows that the reliability values of Web service compositions have some regularity with the number of Web services and the reliability range of transitions. If the FRCPN scale is larger and the minimum value of the reliability range of transitions is smaller, then the reliability value of

FRCPN is smaller, so we need to proper control the scale of Web services in FRCPN, and improve the reliability of a single transition. Next, we can carry out further research in the following areas:

(1) Study the transaction model of Web service compositions and the exception recovery mechanism of Web service compositions, to facilitate the fallback handling for the jobs in the SCP prototype.
(2) Carry out research into the structural verification algorithm of FRCPN to verify the validity of the structure.
(3) Create a comprehensive summary of existing research works to form the theory and application system behind service compositions in the cloud platform, including modeling, searching, matching, automatic composition, validation, transaction handling, exception handling, etc.

Acknowledgments: This work was supported by the National Natural Science Foundation of China (61174140, 61472127, 61272395); National Key Technology Support Program of China Grant No.2012BAH09B02.

Author Contributions: Ziyun Deng put forward the ideas of research work and algorithm design; Lei Chen and Tingqing He performed the experiments; Tao Meng analyzed the data; Lei Chen and Tingqing He contributed tools; Ziyun Deng and Tingqing He wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, B.; Chai, X.; Zhang, L.; Hou, B.; Lin, T.Y.; Yang, C.; Xiao, Y.; Xing, C.; Zhang, Z.; Zhang, Y.; et al. New advances of the research on cloud simulation. In *Advanced Methods, Techniques, and Applications in Modeling and Simulation*; Springer: New York, NY, USA, 2014; Volume 4, pp. 1597–1606.
2. Juan, Y.; Gang, G. Design a new manufacturing model: Cloud manufacturing. In *Proceedings of the 2012 International Conference on Cybernetics and Informatics*; Springer: New York, NY, USA, 2014; Volume 14, pp. 1597–1606.
3. Zhang, L.; Luo, Y.; Tao, F.; Li, B.; Ren, L.; Zhang, X.; Guo, H.; Cheng, Y.; Hu, A.; Liu, Y. Cloud manufacturing: A new manufacturing paradigm. *Enterp. Inf. Syst.* **2014**, *8*, 167–187. [CrossRef]
4. Ren, L.; Zhang, L.; Tao, F.; Zhao, C.; Chai, X.; Zhao, X. Cloud manufacturing: From concept to practice. *Enterp. Inf. Syst.* **2015**, *9*, 186–209. [CrossRef]
5. Lu, Y.; Xu, X.; Xu, J. Development of a hybrid manufacturing cloud. *J. Manuf. Syst.* **2014**, *33*, 551–566. [CrossRef]
6. Cho, S.-H. CAE services on cloud computing platform in South Korea. In *AsiaSim 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 440–446.
7. Yu, J.; Cha, J.; Lu, Y.; Xu, W.; Sobolewski, M. A CAE-integrated distributed collaborative design system for finite element analysis of complex product based on SOOA. *Adv. Eng. Softw.* **2010**, *41*, 590–603. [CrossRef]
8. Wahab, O.A.; Bentahar, J.; Otrok, H.; Mourad, A. Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game. *IEEE Trans. Serv. Comput.* **2016**. [CrossRef]
9. Lee, S.; Kim, H.-G.; Jung, H.; Lee, M.; Song, S.; You, B.-J. OntoPipeliner: An ontology-based automatic semantic service pipeline generator. *Expert Syst. Appl.* **2011**, *38*, 9472–9482. [CrossRef]
10. Angelo, F.; Eugenio, Z. Context-aware composition of semantic web services. *Mob. Netw. Appl.* **2014**, *19*, 235–248.
11. Lin, S. Research on Semantic Similarity Computation and Applications. Ph.D. Thesis, Shangdong University, Shangdong, China, October 2009.
12. Xia, Y.; Liu, Y.; Liu, J.; Zhu, Q. Modeling and performance evaluation of BPEL processes: A stochastic-petri-net-based approach. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2012**, *42*, 503–510. [CrossRef]
13. Du, Y.; Li, X.; Xiong, P. A petri net approach to mediation-aided composition of web services. *IEEE Trans. Autom. Sci. Eng.* **2012**, *9*, 429–435. [CrossRef]
14. Du, Y.; Tan, W.; Zhou, M. Timed compatibility analysis of web service composition: A modular approach based on petri nets. *IEEE Trans. Autom. Sci. Eng.* **2014**, *11*, 594–606. [CrossRef]
15. Yu, W.; Yan, C.; Ding, Z.; Jiang, C.; Zhou, M. Modeling and validating e-commerce business process based on petri nets. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 327–341. [CrossRef]

16. Xia, Y.; Luo, X.; Li, J.; Zhu, Q. A petri-net-based approach to reliability determination of ontology-based service compositions. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 1240–1247.
17. Feng, L.; Masanao, O.; Takashi, K.; Kobayashi, K. A learning fuzzy petri net model. *IEEE Trans. Electr. Electron. Eng.* **2012**, *7*, 274–282. [CrossRef]
18. Chen, L.; Zhang, G. A petri net approach to reliable execution for web service composition. In Proceedings of the 2013 9th International Conference on Computational Intelligence and Security (CIS), Sichuan, China, 14–15 December 2013; pp. 105–109.
19. Nematzadeh, H.; Motameni, H.; Mohamad, R.; Nematzadeh, Z. QoS Measurement of workflow-based web service compositions using colored petri net. *Sci. World J.* **2014**, *2014*, 1–15. [CrossRef] [PubMed]
20. Vidal, J.C.; Lama, M.; Bugarín, A. Toward the use of petri nets for the formalization of OWL-S choreographies. *Knowl. Inf. Syst.* **2012**, *32*, 629–665. [CrossRef]
21. Zhu, L.; Sun, L. Hierarchical colored petri nets based modeling and analyzing for web service composition. *Key Eng. Mater.* **2011**, *467*, 1206–1211. [CrossRef]
22. Li, B.; Ji, S.; Qiu, D.; Cai, J. Generating test cases of composite services based on owl-s and eh-cpn. *Int. J. Softw. Eng. Knowl. Eng.* **2010**, *20*, 921–941. [CrossRef]
23. Xia, Y.; Zhang, X.; Luo, X.; Zhu, Q. Modelling of ontology-based service compositions using petri net. *Elektron. Elektrotech.* **2013**, *19*, 75–78. [CrossRef]
24. Xia, Y.; Wan, N.; Dai, G.; Luo, X.; Sun, T. A non-markovian stochastic petri net-based approach to performance evaluation of ontology-based service composition. *Concurr. Comput. Pract. Exp.* **2012**, *24*, 2255–2267. [CrossRef]
25. Wahab, O.A.; Bentahar, J.; Otrok, H.; Mourad, A. A survey on trust and reputation models for Web services: Single, composite, and communities. *Decis. Support Syst.* **2015**, *74*, 121–134. [CrossRef]
26. Hang, C.-W.; Kalia, A.K.; Singh, M.P. Behind the curtain: Service selection via trust in composite services. In Proceedings of the 2012 IEEE 19th International Conference on Web Services (ICWS), Honolulu, HI, USA, 24–29 June 2012; pp. 9–16.
27. Stantchev, V. *Effects of Replication on Web Service Performance in WebSphere*; International Computer Science Institute: Berkeley, CA, USA, 2008.
28. Stantchev, V.; Malek, M. Translucent replication for service level assurance. In *High Assurance Services Computing*; Springer: New York, NY, USA, 2009; pp. 1–18.
29. Werner, M.; Richling, J.; Milanovic, N.; Stantchev, V. Composability concept for dependable embedded systems. In Proceedings of the International Workshop on Dependable Embedded Systems at the 22nd Symposium on Reliable Distributed Systems (SRDS 2003), Florence, Italy, 5 October 2003; pp. 20–25.
30. Haddad, J.E.I.; Manouvrier, M.; Rukoz, M. TQoS: Transactional and QoS-aware selection algorithm for automatic web service composition. *IEEE Trans. Serv. Comput.* **2010**, *3*, 73–85. [CrossRef]
31. Fan, G.; Yu, H.; Chen, L.; Liu, D. Petri net based techniques for constructing reliable service composition. *J. Syst. Softw.* **2013**, *86*, 1089–1106. [CrossRef]
32. Kim, Y.; Choi, J.-S.; Shin, Y. Trustworthy service discovery for dynamic web service composition. *KSII Trans. Internet Inf. Syst.* **2015**, *9*, 1260–1281.
33. Silic, M.; Delac, G.; Srbljic, S. Prediction of atomic web services reliability for QoS-aware recommendation. *IEEE Trans. Serv. Comput.* **2015**, *8*, 425–438. [CrossRef]
34. Xu, J.; Yao, S. Reliability of SOA systems using SPN and GA. In Proceedings of the IEEE 10th World Congress on Services, Anchorage, AK, USA, 27 June–2 July 2014; pp. 370–377.
35. Garg, H. Reliability analysis of repairable systems using petri nets and vague Lambda—Tau methodology. *ISA Trans.* **2013**, *52*, 6–18. [CrossRef] [PubMed]
36. Gan, M.; Wang, S.; Zhou, M.; Li, J.; Li, Y. A survey of reachability trees of unbounded petri nets. *Acta Autom. Sin.* **2015**, *41*, 686–693.